MacTech

A G A Z I N E

. 11, No. 5 • May 1995

How to establish yourself on the Internet.

NSIDE:

General Magic

Magic Cap™

32128 74887 8 05

$5.85 US
$6.95 Canada
ISSN 1067-8360
Printed in U.S.A.

Want to unlock the power of PowerPC™ microprocessors?

Here's the key.

If you want to crank out code that really screams on PowerPC microprocessor-based systems, then it helps to work with the experts who know the PowerPC Architecture™ inside and out—Motorola.

Motorola's Software Development Kits for Power Macintosh™ leverage our experience as one of the creators of the PowerPC Architecture. We know how to get the highest possible performance out of every member of the PowerPC family, including the PowerPC 603™ and PowerPC 604™ microprocessors. And we build that expertise into our highly optimizing C/C++ and FORTRAN compilers. They'll streamline your code, so your applications run cleaner and faster. And that can give them the edge in the marketplace.

And now, when you license a Motorola C/C++ or FORTRAN SDK for only $349,* you'll get the complete Apple® MPW environment, as well as free upgrades for a full year.

So contact Motorola for more information, or to order your own Macintosh® SDK. And unlock the power inside PowerPC microprocessors.

Call 1-800-347-8384 or 1-512-891-2999.

Internet: ppcinfo@pets.sps.mot.com
WWW: http://www.mot.com/PowerPC
Applelink: motosoftware@applelink.apple.com
*Suggested list price in U.S. dollars. Subject to change.

PowerPC™ Mac™ OS MOTOROLA

Eddy Award Winner** for **Best New Developer Tool
– MacUser Editors Choice Awards, 1993

"A distinct improvement over ResEdit."
– MacTech / MacTutor

"Resorcerer's data template system is amazing!"
– Bill Goodman, author of Compact Pro

"Nuke ResEdit! Resorcerer is mission-critical for us."
– Dave Winer, Userland Frontier

"The color pixel editors are wonderful! A work of art!"
– Dave Winzler, author of Microseeds Redux

"Every Macintosh developer should own a copy of Resorcerer."
– Leonard Rosenthol, Aladdin Systems

"Resorcerer will pay for itself many times over in saved time and effort."
– MacUser review

"The template that disassembles 'PICT's is awesome!"
– Bill Steinberg, author of Pyro! and PBTools

"Resorcerer proved indispensible in its own creation!"
– Doug McKenna, author of Resorcerer

"...a wealth of time-saving tools."
MacUser Review, Dec. 1992

RESORCERER®

Version 1.2.4

The Resource Editor for the Macintosh Wizard

ORDERING INFO

Needs: ≥Mac Plus, ≥ Sys 4.2, 1MB
Likes: ≥Mac Plus, ≥ Sys 7.0, 2MB
32-bit clean, AU/X compatible

Price: $256 (decimal)
(Educational, quantity, or other discounts available)

Includes: 500 page manual
60-day Money-Back Guarantee
Domestic UPS ground shipping

Payment: Check, PO's, or Visa/MC

Extras (call us):
COD, FedEx, UPS Blue/Red, International Shipping

Downloadable Demos/Updaters:
AppleLink: Software Sampler
AOL: Software Libs/Development
CompuServe: MACDEV/Tools
or call us.

New 1.2 Features:

- **New 'cicn', 'ppat', 'crsr', 'acur', 'pltt', 'clut' editors**
- **Powerful icon family editing (all 9 icon types)**
- **Color pixel anti-aliasing, dithering, and lots more**
- **Complete 'PICT' disassembly and reassembly**
- **Resource sorting; ROM resource browsing**
- **120 template field parsing types now supported**
- **New insertion & deletion template field types**
- **Text-only 'PICT' resources**
- **Lots of improvements throughout**

- Easier, faster, more Mac-like, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Help, AppleEvent, OCE, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Well-designed, helpful developer tools being added all the time
- Relied on by thousands of Macintosh developers around the world

MATHEMÆSTHETICS, INC.
P.O. Box 298 • Boulder • CO • 80306-0298 • USA
Phone: (303) 440-0707 • Fax: (303) 440-0504
AppleLink/AmericaOnline: RESORCERER • Internet: resorcerer@aol.com

For Macintosh Programmers & Developers

MacTech MAGAZINE

A publication of XPLAIN CORPORATION

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?** If you have any questions, feel free to call us at 310/575-4343 or fax us at 310/575-0925.

DEPARTMENTS	Internet	eWorld	CompuServe	AppleLink	America Online
Orders, Circulation, & Customer Service	custservice@xplain.com	MT.CustSvc	71333,1063	MT.CUSTSVC	MT CUSTSVC
Editorial	editorial@xplain.com	MT.Editors	71333,1065	MT.EDITORIAL	MT EDITORS
Programmer's Challenge	progchallenge@xplain.com	MT.PrgChal	71552,174	MT.PROGCHAL	MT PRGCHAL
Ad Sales	adsales@xplain.com	MT.AdSales	71552,172	MT.ADSALES	MT ADSALES
Online Support	online@xplain.com	MT.Online	–	–	–
Accounting	accounting@xplain.com	accounting@xplain.com	–	–	
Marketing	marketing@xplain.com	MT.Mktg	–	–	–
Press Releases	pressreleases@xplain.com	pressreleases@xplain.com	–	–	–
General	info@xplain.com	info@xplain.com	71333,1064	MACTECHMAG	MacTechMag
Online support area	ftp://ftp.netcom.com/pub/xp/xplain	*use shortcut:* MACTECH	*type* GO MACTECHMAG	*see* Third Parties: Third Parties (H-O)	*use keyword:* MACTECHMAG

Authors & Regular Contributors

MacTech Magazine is grateful to the following individuals who contribute on a regular basis. We encourage others to share the technology. We are dedicated to the distribution of useful programming information without regard to Apple's developer status. For information on submitting articles, ask us for our **writer's kit** which includes the terms and conditions upon which we publish articles.

Richard Clark
General Magic
Mountain View, CA

David R. Mark
M/MAC
Arlington, VA

Mike Scanlin
Programmer's Challenge
Mountain View, CA

Chris Espinosa
Apple Computer, Inc.
Cupertino, CA

Jordan Mattson
Apple Computer, Inc.
Cupertino, CA

Symantec Technical Support Group
THINK Division
Eugene, OR

Printed on recycled paper.

PRINTED WITH SOY INK

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 1617 Pontius Avenue, 2nd Floor, Los Angeles, CA 90025-9555. Voice: 310/575-4343, FAX: 310/575-0925. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Second Class postage is paid at Los Angeles, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 250055, Los Angeles, CA 90025-9555.

TABLE OF CONTENTS

May 1995 • Volume 11, No. 5

For Macintosh Programmers & Developers

MacTech MAGAZINE

THE EDITOR'S VIEWPOINT

By Scott T Boyd, Editor
http://www.hax.com/

To Web or not to Web, was that the question? I've encountered two positions on this lately. The first holds that only a year or two stands between our current state and a world where developers take Internet access for granted. The second position holds that we're a long way from a majority of us having Internet access.

I'm betting on a wired world of developers. How can I be so certain? Well, I'm not, but I have this very strong memory of a time, about ten years ago, when I saw something new and said to myself, "This is cool!" Almost every single dollar I've earned since then derived from my interest in exploring Macintosh. I delved, learned, explored, and played to the point where my father asked whether I might not be limiting my career choices. I suppose I did. I see another such choice forming up right about now.

How can I tell? Oh, I don't know, maybe it's just that I've fed all of my spare time to the Internet like quarters into a great arcade game. I've set up a domain name server, configured a mail system, and added accounts and aliases to a unix machine so mail goes to the right places. I've installed a gateway and a router so my wife can do e-mail and netnews. I've set up mailing lists, set up dial-in service, and helped friends get their homes and offices onto the Net. I've fought (endlessly?) to convince my modems and router to keep my office connected to my Internet provider. And lately I've poured a lot of time into building all sorts of web pages. Along the way, I've picked up a couple of things I'd like to share with you.

While the telephone system connects almost all of us to each other, it restricts how we interact. For example, if I called your office to see who answered the phone, only to hang up immediately when an interesting person didn't answer, you might consider that rude. On the other hand, no one minds at all if I flit in, take a look at a web page, and flit away. Likewise, you might not want to answer your office phone at 4AM, nor might you want to pay someone to wait by the phone for the occasional 4AM call. Web servers, to everyone's benefit, don't get sleepy, and they don't mind waking up in the middle of the night.

People who cruise the World Wide Web enjoy sitting in the driver's seat (even at 4AM). Set up your web site with this in mind. They're driving, so give them what they might be looking for. Let them decide how much, when, and in what order they'll check out what you've published. Don't count on them calling you if they have any questions.

Late last week I decided to buy a router. Between living on the Left Coast and staying up until all hours, I rarely get enough spare phone time in before many of the places I need to call close for the day. That's one reason I love shopping on the Web – it never closes. So, at 4AM, I started looking for routers. I came across a promising site (http://www.rockwell.com). I dug around for a while, and finally found a router that grabbed my interest. Hey, no prices! They didn't even list the protocols it supports.

Don't Do #0 – Don't leave out information that buying customers need to make buying decisions (*all* web-site visitors are buying customers, by the way; you just have to show them something they want to buy). Fortunately, they included an e-mail URL, so I dropped them a note saying, "Please get me the following info and I might buy one right now." I figured that I might have the info I needed when I woke up later in the day. Sure enough, I had mail from them. This leads to…

Don't Do #1 – Don't send e-mail to a customer saying, "Please send us your geographical location so we can have a salesperson from your regional sales office call you." Momentarily dumbfounded, I knew what I had to do. In keeping with the true spirit of directness on the Internet, I dropped them a little note and suggested that maybe they weren't ready for the Internet (Another thing I love about the Net – one mouse-click and no more pushy salesperson). This leads to **Do #0** – respond to such mail quickly and with attention to details on the questions the sender asked. It doesn't hurt to go further and apologize for having an incomplete web site, and to offer to fix the specific problems the user had with it. That's what Rockwell did, and I'm now the happy owner of a Rockwell NetHopper.

One Last Shameless MacHack Plug

MacHack X (June 22-24) is right around the corner. Got your hack done yet? To get more info on the only marketing-hostile, programmer-centric conference of the year, check out http://www.hax.com/HackContest and call or e-mail Expotech at (313) 882-6942 and expotech@aol.com.

Let's Take A Stand

While we're on the subject of paradigm-altering technologies, we'll take the opportunity to **endorse OpenDoc**. The tools are inadequate, the software hasn't shipped to customers yet, and there's a *lot* to learn. Nevertheless, it's a superior technology, and the one we're picking. Watch for increased coverage of OpenDoc here in these pages in the months to come.

Food For Thought

The only way to deal with bureaucrats is with stealth and sudden violence. – *UN Secretary-General Boutros Boutros-Ghali*

I think of George Orwell… And I remember the commercial, "**Why 1984 won't be like 1984…**" and I say to myself, "Maybe 1984 was just a few years early!" – *Allan Foster*

You can kill a project, but you can't kill the past.
– *A much-revered DTS engineer, speaking on job security*

There are Billions of reasons to protect your software

You only need one reason to protect with Sentinel:

It's the worldwide standard in software protection.

Piracy is the greatest threat to the world's software industry. Developers lose billions in sales to software piracy each year. Protect your software and get all the revenue you deserve.

More developers rely on Sentinel®, from Rainbow, than any other software protection in the world.

And for good reason. Sentinel performs where it matters most: leading the industry in technology, quality, reliability and support.

So when it's time to protect your Macintosh application - and you need just-in-time delivery - protect with Sentinel. Protect with confidence.

Call Rainbow for a Sentinel Developer's Kit and a FREE copy of *"The Sentinel Guide to Securing Software."*

Mac OS

ISO 9002 CERTIFIED

Call Today!
1-800-852-8569

RAINBOW TECHNOLOGIES

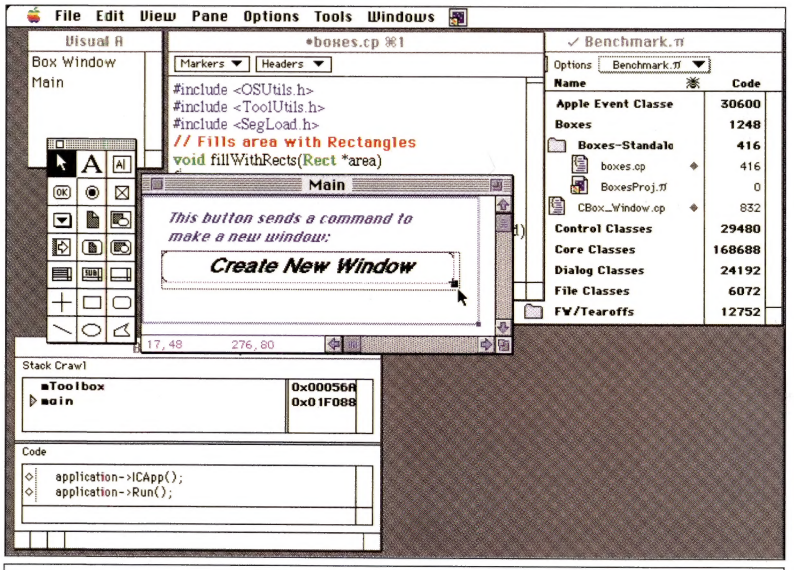
SENTINEL
Securing the future of software

9292 Jeronimo Road • Irvine, CA 92718 • Tel: 714/454-2100 • Fax: 714/454-8557 • *Applelink D3058* •*International offices:*
U.K. (44) 1932 570066 • France (33) 1 47 38 21 21 • Germany (49) 89 32 17 98 0 • North Carolina 800/843-0413

It's the highly-regarded industry standard, used by more developers than any other Macintosh development system. And now it's been totally re-engineered for Power Mac.

Introducing Symantec C++™ 8.0 for native Power Mac.

Not since the original THINK C™ has anything so dramatically boosted your productivity. There's a Visual Architect™ to generate GUI code instantly. An advanced Project Manager to handle the largest and most complex jobs. Plus native Power Mac tools for radically improved performance.



Boost your productivity with our Visual Architect, Advanced Editor, Debugger and Project Manager.

THE INDUSTRY STANDARD JUST MOVED TO A HIGHER POWER. SYMANTEC C++ 8.0 FOR POWER MAC.

DRAW ON THE INDUSTRY STANDARD.

With Symantec C++ 8.0, you simply draw the user interface – including windows, dialogs, controls, icons and menus. Then the built-in Visual Architect generates the code with the click of a mouse. Now you can spend more time on what really sets your application apart – its functionality.

A HIGHER STANDARD FOR SPEED.

The new high-performance compiler is dramatically faster than the previous version, so you can become more productive than ever.

And for even more power, we've added an Advanced Project Manager. It gives you drag and drop so you can easily add files, Named Option Sets for changing complex sets of options fast, plus support for even the largest applications.

There's also a new editor and a browser for modifying and navigating source files, a new debugger, an incremental linker, THINK Class Library™ 2.0 and much more.

NEW! FAST C AND C++ COMPILERS
for both Mac and Power Mac

NEW! VISUAL ARCHITECT
for the Power Mac

NEW! THINK CLASS LIBRARY
2.0 *for Power Mac*

NEW! POPUP MENUS
take you right to declarations & headers

NEW! DEBUGGER
for nested projects and shared libraries

NEW! PROJECT MANAGER
for multiple targets and hierarchical support

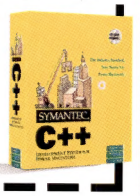
NEW! SPLIT PANE EDITOR
for syntax highlighting and auto formatting.

A DOUBLE STANDARD FOR 68K AND POWER MAC.

Symantec C++ 8.0 is the first native Mac/Power Mac development system to support C++ templates, nested classes and multiple inheritance, as well as ANSI C. Plus 8.0 includes version 7.0 for 68K support.

This double standard gives you everything you need for both 68K and Power Mac development.

To order at a special upgrade price of $149.95, call 1-800-628-4777 Ext. 9H21. Be sure to ask about the Symantec Developer's Advantage Program for premium support and regular updates. Or visit your local software store.



SYMANTEC.®

Price good in U. S. only. For more information in Canada, call 1-800-667-8661, ext. 5513. In Australia, call 2-879-6577. In Europe, call 31-71-353111. Symantec C++, Visual Architect, THINK Class Library are trademarks of Symantec Corporation. All other trademarks are the property of their respective holders.© 1995 Symantec Corporation. All rights reserved.

By Dave Mark, MacTech Magazine Regular Contributing Author

Sprocket Menus, Part 1

My February '93 Getting Started column featured a program called MenuMaster. MenuMaster constructed a menu bar consisting of four menus: The traditional , **File**, and **Edit** menus, as well as a special **Options** menu (Figure 1). Selecting the first item changes it from **Change My Name** to **Change Me Back** Again. Selecting the first item again changes it back to **Change My Name**.

Selecting **Disable Me** disables the second item and enables the third item. If you then select the newly enabled **Enable Previous Item**, it gets disabled and **Disable Me** is reenabled.

If you select **Add Extra Menu**, a new menu is inserted in the menu bar and **Add Extra Menu** is disabled. The new menu, titled **Extra Menu**, features a single item, **Delete Me**. Selecting **Delete Me** deletes the extra menu from the menu bar and reenables **Add Extra Menu**.

Finally, selecting **Append Item** adds an extra item (**Can't Delete Me...**) to the end of the menu. As its names implies, there's no way to delete this extra item.

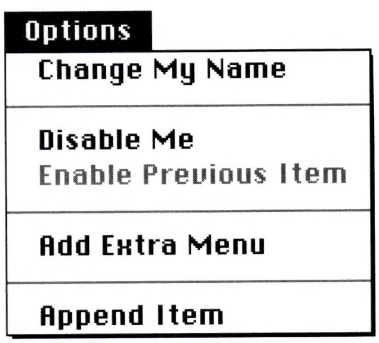


Fig. 1. MenuMaster's **Options** *menu.*

A Sprocket Version of MenuMaster

This month we're going to use Sprocket to implement most of MenuMaster's functionality. We'll skip the ability to append an item to the end of a menu for two reasons. First, appending a single item to the end of a menu just isn't done that often and isn't particularly useful. More importantly (and probably for the same reason), Sprocket doesn't give you an easy way to append a new item to a menu.

> If you come up with a good reason to add this functionality to Sprocket (or if you have any comments or bugs to report), send e-mail to `sprocket@hax.com`.

As I mentioned last month, Sprocket based its menu-handling model on that used by OpenDoc. At the heart of this model is a replacement for the MENU resource type. A CMNU resource is just like a MENU resource, with one important addition. Each menu item features a command number. You'll use this command number to refer to the item, instead of the more traditional method of specifying the menu the item belongs to, along with the item's position in the menu (e.g., menu 129, item 4).

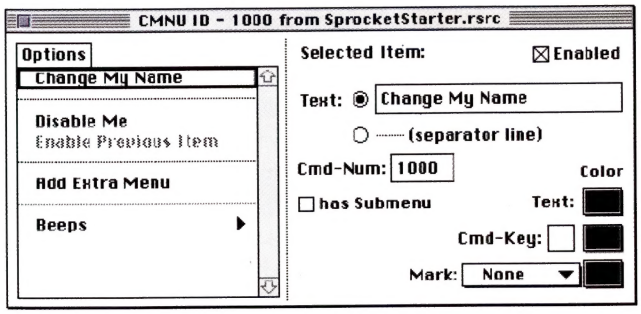


Figure 2. The CMNU resource, featuring a **Cmd-Num** *field for each menu item.*

Check out the ResEdit snapshot in Figure 2. It shows the CMNU resource that represents our new **Options** menu. The first menu item, **Change My Name**, is selected. The command number for this item is 1000. When the user selects this item, Sprocket will pass the associated command number (in this case, 1000) as a parameter to the routine `HandleMenuCommand()` (it's in the file `SprocketStarter.cp`). Instead of creating a separate item dispatch routine for each menu (`HandleAppleMenu()`,

HandleFileMenu(), etc.), you'll create a single switch statement containing cases for all your commands.

Sprocket automatically creates a menu bar at application startup. In C++ terms, Sprocket constructs a TMenuBar object, which is implemented in the files TMenuBar.cp and TMenuBar.h. Here's the TMenuBar class definition:

```
class  TMenuBar
  {
public:
                                        Resource ('MBAR' and 'CMNU') Utilities
  OSErr           GetNewMenuBar(short whichMBAR);
  MenuRef         GetMenuFromCMNU(short whichMenu);
                                        Menu command mapping functions
  MenuCommandID   GetCommand(MenuID menu, MenuItemID item);
  void    GetMenuAndItem(MenuCommandID commandNum,
            MenuID * returnedMenu, MenuItemID * returnedItem);
  OSErr   RegisterCommand(  MenuCommandID commandNum,
                            MenuID menu, MenuItemID item);
  OSErr   UnregisterCommand(MenuCommandID commandNum);
                                        Menu enable/disable routines for menu items
  void    EnableCommand(MenuCommandID commandNum,
                        Boolean enable);
  void    EnableAndCheckCommand(MenuCommandID commandNum,
                        Boolean enable, Boolean check);
  void    GetItemString(MenuCommandID commandNum,
                        StringPtr itemString);
  void    SetItemString(MenuCommandID commandNum,
                        StringPtr itemString);
                                        helpful utility functions
  void          HideMenuBar();
  void          ShowMenuBar();
  void          RedrawIfNeeded();
  void          Invalidate();
  void          Validate();

private:
                                        "globals"
  static Boolean      fgMenuBarNeedsRedraw;
  static Boolean      fgMenuBarHidden;
                                        mapping tables
  TMenuCommandTable   fCommandTable;
  TMenuItemTable      fMenuItemTable;
                                        internal methods
  MenuHandle GetMenuHandleAndItemFromCommand(
              MenuCommandID commandNum,
              MenuID *menu,MenuItemID *item);
  };
```

The first member function, GetNewMenuBar() uses the specified MBAR resource to build a new menu bar. Though this version of Sprocket only creates a single menu bar, this might not be the case in the future. For now, a pointer to the menu bar object is stored in the global gMenuBar. Take a minute to open up the file SprocketMain.cp and check out the code around line 363. This is where Sprocket creates the TMenuBar object based on the MBAR resource in SprocketStarter.rsrc.

The member function GetMenuFromCMNU() loads a CMNU resource and walks through it, one item at a time. It builds a traditional menu structure, passing each item's command number to the RegisterCommand() member function, which adds the command to Sprocket's menu command table. If you are going to take advantage of Sprocket's menu command mechanism, you must register your menu item commands. If you base your menus on a CMNU resource, GetMenuFromCMNU() will register your menu items automatically. If the menus in your MBAR resource correspond to a CMNU resource, Sprocket will register the menu items automatically.

If you don't want to use a CMNU resource, you can still add and delete your menus to and from the global menu bar yourself. For example, since a font or size menu will have a dynamic number of items, the CMNU resource just doesn't make sense. We'll look at that process in a future column.

The member function GetCommand() takes a menu and item ID and returns the associated command. GetMenuAndItem() takes a command and returns the associated menu and item ID.

If you want to delete a menu whose commands have been registered, you can use the UnregisterCommand() member function to, one-at-a-time, unregister the commands in that menu. Otherwise, you'll orphan commands in the command table.

EnableCommand() and EnableAndCheckCommand() let you enable, disable, check, and uncheck a menu command. GetItemString() and SetItemString() allow you to retrieve and set an items name using its command.

HideMenuBar() and ShowMenuBar() let you hide and show the menu bar (what a concept!). Invalidate() marks the menu bar as needing to be redrawn. Validate() sets the menu bar as up to date. RedrawIfNeeded() redraws the menu bar if the invalid flag has been set. Note that RedrawIfNeeded() is called in Sprocket's main event loop, so there's no need for you to call it yourself.

This Month's Resources

Sprocket gets its resources from four different resource files. CreditsBox.rsrc contains the resources used to build the Sprocket about box. StandardMenus.rsrc contains some standard MENU and CMNU resources. If you want to change any of these menus, copy the appropriate resource from StandardMenus.rsrc into SprocketStarter.rsrc and delete the original from StandardMenus.rsrc. Modify the version you copied into SprocketStarter.rsrc.

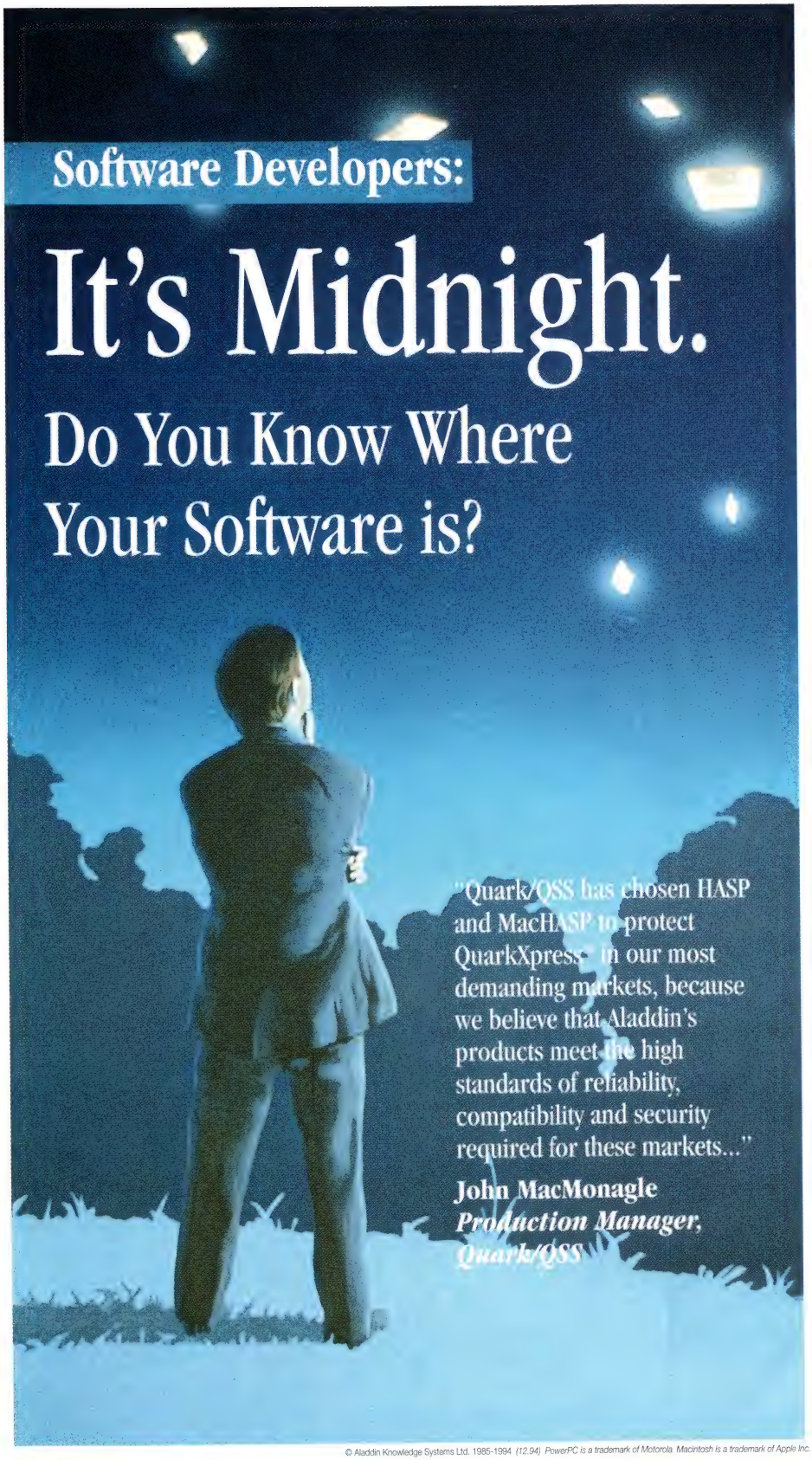
Sprocket.rsrc contains various resources used by Sprocket and should not be modified. SprocketStarter.rsrc is your resource center. Put all the resources you add to Sprocket there.

You'll need to modify one resource and add three new ones to SprocketStarter.rsrc. First, open up MBAR 128 and add menu ID 1000 to the list already in place.

> If you're not using Projector (the source code control system), you might want to delete the ckid resources you'll find in each of the resource files. That will get rid of the annoying link error complaining about the multiply-defined resource.

Next, you'll create your three CMNU resources. The first represents the **Options** menu we want to add to the end of the menu bar. In general, when you add a new resource to Sprocket, you'll start numbering your resources from 1000, instead of at 128 the way you normally would. This is just a convention, and might change as Sprocket grows up.

> When you create CMNU 1000, be sure to change the resource ID in both places: once in the "Get Info" box and also in the "Edit Menu and MDEF ID" dialog.



Bringing software into the world is a little like bringing up children. You always know where they start, but you seldom know where they'll end up. These days, with illegal use of software so common, concerned developers have good reason to worry about the products of their labor. That's where MacHASP comes in.

Like a responsible babysitter, MacHASP accompanies your software wherever it goes. With MacHASP there, your software won't run out of control. Without MacHASP, in fact, your software won't run at all.

For developers, MacHASP provides the highest level of security and reliability. For legitimate users, MacHASP is a friendly and transparent solution. Once connected, they won't even feel it's there.

And if your child wants to play with its friends, a single Net-MacHASP lets it run free around a local area network. But always under your supervision and control.

Get serious about software protection. Since 1984, nearly one million HASP keys have enabled thousands of PC & Mac software developers, in more than 60 countries, to protect their software. To find out why MacHASP is considered the best product in the market, order your MacHASP Developer's Kit today.

1-800-223-4277

ALADDIN

The Professional's Choice

North America	**Aladdin Software Security Inc.** Tel: (800) 223 4277, 212-564 5678 Fax: 212-564 3377 E-mail: sales@hasp.com
Intl Office	**Aladdin Knowledge Systems Ltd.** Tel: 972-3-537 5795, Fax: 972-3-537 5796 AppleLink: ALADDIN.KNOW E-mail: aladdin@aladdin.co.il
United Kingdom	**Aladdin Knowledge Systems UK Ltd.** Tel: 0753-622266, Fax: 0753-622262
France	**Aladdin France SA** Tel: 1 40 85 98 85, Fax: 1 41 21 90 56

member of SPA

■ **Australia** Conlab 3 8985685 ■ **Benelux** Aladdin Benelux 080 782098 ■ **Czech** Atlas 2 766085 ■ **Chile** Micrologica 2 222 1388
■ **Denmark** Berendsen 39 577100 ■ **Egypt** Zeineldein 2 3604632 ■ **Finland** ID-Systems 0 870 3520 ■ **Germany** CSS 201 278804
■ **Greece** Unibrain 1 6856320 ■ **Italy** Partner Data 2 26147380 ■ **Japan** Athena 3 58 213284 ■ **Korea** Dae-A 2 848 4481 ■ **Mexico** SiSoft 5 5439770
■ **New Zealand** Training 4 5666014 ■ **Poland** Systherm 61 480273 ■ **Portugal** Futurmatica 1 4116269 ■ **South Africa** D Le Roux, 11 886 4704
■ **Spain** PC Hardware, 3 4493193 ■ **Switzerland** Opag 61 7169222 ■ **Taiwan** Teco 2 555 9676 ■ **Turkey** Mikrobeta 312 467 7504

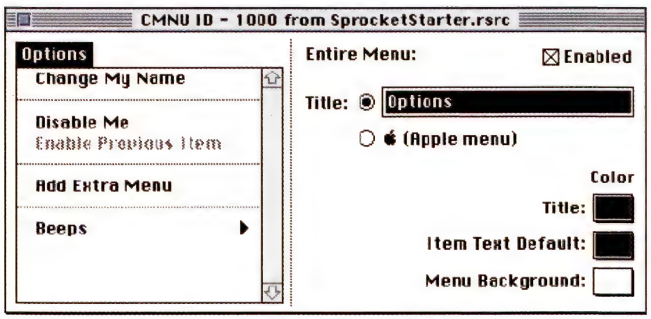


Figure 3. CMNU 1000

Enter a command number of 1000 for the item **Change My Name**, 1001 for **Disable Me**, 1002 for **Enable Previous Item**, 1003 for **Add Extra Menu**, and 1004 for **Beeps**. Next, disable the item **Enable Previous Item**. After that, click on the **Beeps** item, check the **has SubMenu** checkbox and enter 100 as the submenu ID (Figure 4). Since submenu IDs are limited to a single byte, we won't be able to give the submenu CMNU resource an ID greater than 1000. So much for sticking to conventions!

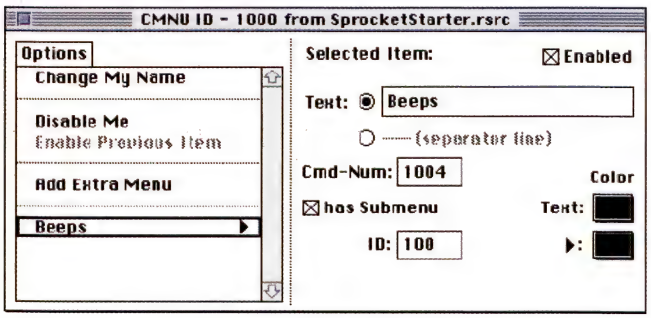


Figure 4. The **Beeps** *item, with its submenu ID of 100 entered.*

Next, create a new CMNU resource with an ID of 1001 (Once again, be sure to change the ID in both places). The menu will have a title of **Extra Menu** and a single item, **Delete This Menu**. Give the item **Delete This Menu** a command of 1007 (Figure 5).

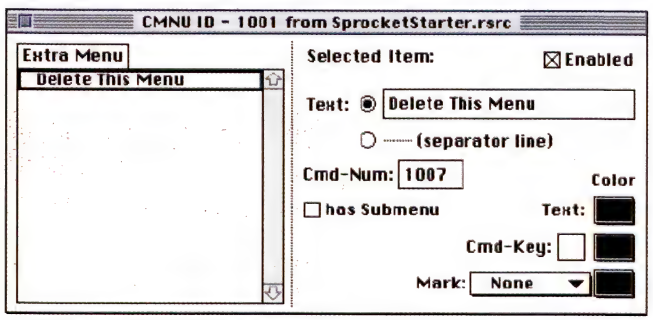


Figure 5. CMNU 1001

Finally, create a CMNU resource with an ID of 100. Add two items, **Beep Once** with a command ID of 1005 and **Beep Twice** with a command ID of 1006 (Figure 6).

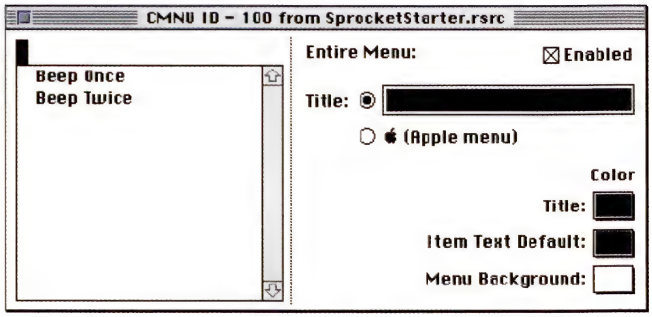


Figure 6. CMNU 100.

Save your changes and quit your resource editor.

Modifying the Source Code

Now launch CodeWarrior or Symantec C++ and edit `SprocketStarter.h`. Start by adding this global reference to the file:

```
extern Boolean    gItemNameChanged;
```

`gItemNameChanged` is a `Boolean` that indicates whether the item **Change My Name** has been selected. It tells us whether the item should read **Change My Name** or **Change Me Back Again**.

Next, add this `enum` to the file:

```
enum
{
  mSubMenu          = 100,
  mExtraMenu        = 1001,

  cChangeName       = 1000,
  cDisableMe        = 1001,
  cEnablePrevious   = 1002,
  cAddExtraMenu     = 1003,
  cBeeps            = 1004,
  cBeepOnce         = 1005,
  cBeepTwice        = 1006,
  cDeleteExtraMenu  = 1007
};
```

The first two constants specify the two CMNU resource IDs. The next 8 specify the menu command IDs. Notice that the menu constants start with a lower case "m" and the commands start with a lower case "c". Unfortunately, the Apple event registry starts all its class names with a lower-case "c", so be on the lookout for name collisions.

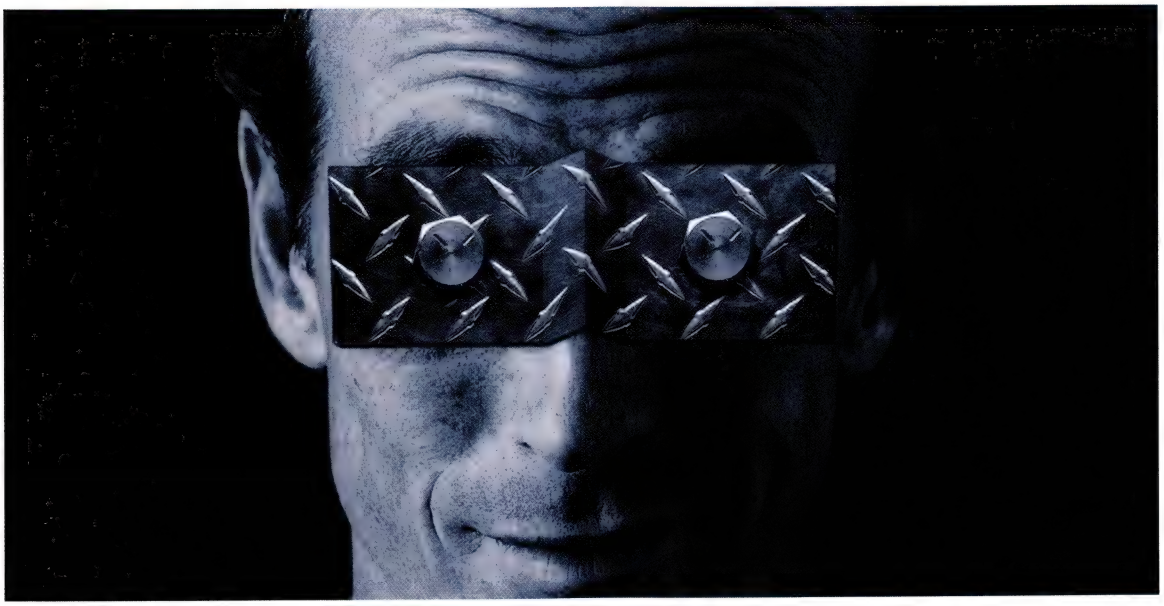
Next, add these three constants to the file:

```
const StringPtr kUnchangedName = "\pChange My Name";
const StringPtr kChangedName = "\pChange Me Back Again";
const short kLastMenu = 0;
```

The first two are just Pascal strings we used for the menu names. We really should have implemented these strings as 'STR ' resources to make the code easier to localize. In general, I try never to specify strings in code, but I guess I was just feeling a bit lazy.

The last constant will be used in our call of `InsertMenu()`, telling `InsertMenu()` to insert the menu at the end of the menu bar.

C/C++ without Object Master



Introducing a powerful new way of looking at code development.

Open your eyes to Object Master™, the most innovative programming tool available on the market today. With its powerful editors and intuitive windows, Object Master gives you the unsurpassed freedom to develop code quickly and accurately.

A Real Eye Opener

Use Object Master's unlimited number of browser windows to access code components and display their definitions, ready for editing. All changes you make in the browser windows are automatically displayed throughout the environment, consistently ensuring current and accurate code. With the browser windows, you can also view a class list displaying the hierarchical relationships between classes, and use pre-made templates to create classes and methods.

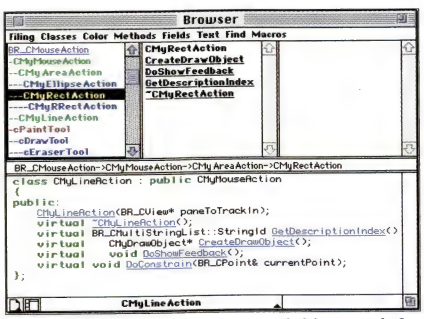
Picture Perfect Code

While the browser windows let you edit specific pieces of code, Object Master's File editor gives you a full-file view of your code, allowing you to quickly perform universal edits.

To help you identify important pieces of code easily, Object Master color codes and formats language elements. With a single keystroke, Object Master will look up parameters for methods or functions contained in the project and paste them directly into your source code.

Improved Insight

Don't worry about the physical location of your code—Object Master parses all files and maintains a data dictionary of project components. The dynamic environment updates your entire project automatically as you edit without compilation!



True browser windows let you see code like never before.

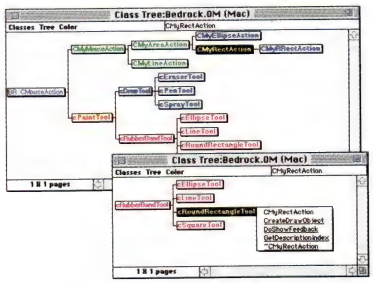
Object Master provides you with all the editing and navigational capabilities you require to write code productively. For example, the Class Tree window graphically displays your project's class hierarchy and allows you to expand and collapse "branches" and open multiple windows displaying specific code components.

Power Macintosh Support

ACI offers two versions of this outstanding programming tool: Object Master and Object Master UNIVERSAL. Both run on the traditional 68K-based Macintosh computer and the new Power Macintosh, taking full advantage of the features of each platform.

Object Master supports C and C++ and works seamlessly with Symantec's THINK Project Manager and Metrowerk's CodeWarrior. Separate versions are available for the 68K-based Macintosh and the new Power Macintosh computers.

Object Master UNIVERSAL supports C, C++, Pascal, Modula-2, and all major compilation systems. It can be installed on both the 68K-based Macintosh and the new Power Macintosh.



"...Object Master pays for itself in a week, even at suggested retail price." –Macworld Magazine

See Object Master for Yourself

Call **(800) 384-0010**, and we'll send you a free demo disk of Object Master—because seeing really is believing.

ACI US Inc. 20883 Stevens Creek Blvd., Cupertino, CA 95014
Tel. 1 408 252 4444. Fax 1 408 252 0831. AppleLink D4444

Next, edit the file SprocketStarter.cp. Start by adding this global definition at the top of the file:

```
Boolean     gItemNameChanged = false;
```

Next, add these three lines to the beginning of the routine SetUpApplication():

```
MenuRef     hierMenu;

hierMenu = gMenuBar->GetMenuFromCMNU( mSubMenu );
InsertMenu( hierMenu, -1 );
```

GetMenuFromCMNU() loads CMNU 100, registers all the commands, and returns a MenuHandle to a standard menu based on the CMNU resource. InsertMenu() inserts the resulting menu in the menu bar.

Finally, add the cases to handle our new commands to the switch in HandleMenuCommand() further down in SprocketStarter.cp. Here's my edited copy of HandleMenuCommand():

HandleMenuCommand

```
void
HandleMenuCommand(MenuCommandID theCommand)
  {
  MenuRef     extraMenu;
  OSErr     err;

  switch (theCommand)
    {
    case cAbout:
      AboutBox();
      break;

    case cNew:
      CreateNewDocument();
      break;

    case cOpen:
      OpenExistingDocument();
      break;

    case cPreferences:
      TPreferencesDialogWindow * prefsDialog =
                          new TPreferencesDialogWindow;
      break;

#if qAOCEAware
    case cNewMailableWindow:
      TMailableDocWindow *aWackyThing = new TMailableDocWindow;
      break;
#endif
```

Here come the new commands. This first one switches the first menu item between **Change My Name** and **Change Me Back Again**. Notice that we're using the global TMenuBar object to change the menus. If Sprocket ever gets modified to use more than one menu bar, we'll have to modify this code to be sure we use the menu bar that contains the menu we want to work with. Of course, if that happens, you can count on some sample code in this column to show you how to do that.

```
    case cChangeName:
      if ( gItemNameChanged )
        gMenuBar->SetItemString( cChangeName, kUnchangedName );
      else
        gMenuBar->SetItemString( cChangeName, kChangedName );
      gItemNameChanged = ! gItemNameChanged;
      break;
```

This command disables **Disable Me** and enables **Enable Previous Item**.

```
    case cDisableMe:
      gMenuBar->EnableCommand( cDisableMe, false );
      gMenuBar->EnableCommand( cEnablePrevious, true );
      break;
```

This command does just the opposite.

```
    case cEnablePrevious:
      gMenuBar->EnableCommand( cDisableMe, true );
      gMenuBar->EnableCommand( cEnablePrevious, false );
      break;
```

This command disables the item that spawned this command in the first place (**Add Extra Menu**), then builds a new menu from the extra menu CMNU resource. We add the new menu to the end of the menu bar, then call Invalidate() to force the menu bar to get redrawn.

```
    case cAddExtraMenu:
      gMenuBar->EnableCommand( cAddExtraMenu, false );
      extraMenu = gMenuBar->GetMenuFromCMNU( mExtraMenu );
      InsertMenu( extraMenu, kLastMenu );

      gMenuBar->Invalidate();
      break;
```

This command deletes the extra menu we added with the previous command. First, we reenable the **Add Extra Menu** item. Notice that we didn't have to retrieve the menu that this item belongs to. All we needed was the command. This definitely makes life a lot simpler.

```
    case cDeleteExtraMenu:
      gMenuBar->EnableCommand( cAddExtraMenu, true );
```

Since the TMenuBar class doesn't support a DeleteCMNU() method, we'll have to deregister the command by hand. A DeleteCMNU() method would step through all the items in the specified menu, calling UnregisterCommand() for each item. It would then delete the menu for us. Since our extra menu only contains a single item, it's no big deal to do this by hand. Once we are done, we'll force a menu bar redraw. Look for a DeleteCMNU() method in a future version of Sprocket.

```
      err = gMenuBar->UnregisterCommand( cDeleteExtraMenu );
      DeleteMenu( mExtraMenu );

      gMenuBar->Invalidate();
      break;
```

This next command corresponds to the parent menu of our hierarchical submenu. Normally, this command will never get called because the menu manager won't detect a selection of the parent item of a submenu (Figure 7). There are times when this is useful, however. For example, imagine if you built a menu of applications, where each application item had a submenu listing some frequently used documents that can be opened by that application (NowMenus does this). If you select a document from a submenu, its parent application gets launched and opens the selected document. If you release the mouse with the application selected (without selecting a document from the submenu), you might want to launch the

DELL OFFERS MACINTOSH® DEVELOPERS THE CHANCE TO OPEN UP A NEW WINDOW.

OPTIPLEX™

THE DELL®
DEVELOPER SYSTEM

DELL® OPTIPLEX™ 590/L
90MHz PENTIUM PROCESSOR

- 528MB Hard Drive/24MB RAM
- 1 MB VRAM
- Microsoft Mouse
- Sound Blaster 16 with Triple Speed CD-ROM & Speakers
- Dell Ultrascan 15" Trinitron
- 3Com EtherLink III Combo
- 3-year limited warranty*
- Microsoft Windows NT Workstation V3.5
- Microsoft MSDN Level II Membership
- Microsoft Visual C++ Subscription

$3,999
Product Code **#300622**

The Dell Developer system will set you up with everything you need to start developing great applications for Windows® 95. With 90 Mhz Pentium® power and speed, Plug-n-Play, advanced power management and much more.

Included is the Microsoft Developer Network Level II subscription which gives you all the information and SDK's that you need for the whole year.

Also included is the Microsoft® Visual C++™ Subscription which gives you the tools you need for a whole year.

With these subscriptions you will get the Windows 95 SDK and Windows 95 specific tools, when they are released to developers.

DELL®

(800) 627-9862 MONDAY-FRIDAY 7AM-9PM CT • SATURDAY 10AM-6PM CT
SUNDAY 12PM-5PM CT • CANADA CALL 800 668-3021

*Prices valid in the US only. Some products and promotions may not be available outside the US. Prices and specifications subject to change without notice. *For a complete description of Dell's 3-year limited warranty, please write to Dell USA L.P., 2214 W Braker Lane Bldg 3, Austin TX 78758-4053 Attn: Warranty. Microsoft and Windows 95 are registered trademarks and Visual C++ is a trademark of Microsoft Corporation in the United States and other countries. Pentium is a trademark of Intel Corporation. Dell disclaims proprietary interest in the marks and names of others. © 1995 Dell Computer Corporation. All rights reserved.*

application without specifying a document.

The point here is this: Specify commands for all your menu items, even the hierarchical parent menus. A future version of Sprocket might include a workaround to execute commands associated with these currently orphaned items.

```
case cBeeps:
  break;
```

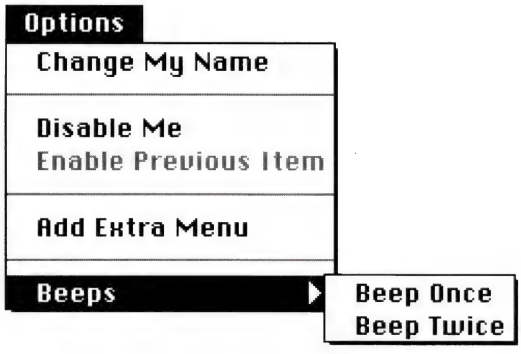


Figure 7. A menu item with its submenu showing. If the mouse button was released at this point, the **Beeps** *item (rather than either of the submenu items) would be selected.*

These next two items are horribly technical. They beep either once or twice, depending on the item selected.

```
    case cBeepOnce:
      SysBeep( 20 );
      break;
    case cBeepTwice:
      SysBeep( 20 ); SysBeep( 20 );
      break;
    default:
      break;
    }
}
```

Till Next Month

There are still some concepts that we need to get into regarding Sprocket and menus. For example, when do you make the decision about which menu items you will enable and disable to ensure that things are set up correctly before a user makes a selection from a menu. How should the frontmost window affect which menu items are enabled or disabled? We'll explore these important issues in next month's column. See you then!

Designed for

Microsoft®
Windows™95

How ya doin'? They say opportunity knocks. This time opportunity is breaking down the door. Windows™ 95 looks to be a phenomenal success, and your Mac® programming experience has prepared you to be one of the best Windows™ 95 developers in the business.

The Scoop! **You love the Mac®, Right?** Because early on, it let you develop cool new applications that you couldn't create anywhere else. Applications that delivered a compelling end-user experience. Applications that worked together more closely than ever before. We know; we feel the same way. That's why Microsoft was one of the first and strongest supporters of the Mac, even before it was launched. Last year we shipped more applications for the Mac than anyone else. We still support the Mac—and you should too.

Because you love the Mac experience you should also look at Windows® 95. Intuitive user interface, Plug-n-Play hardware, long file names…they're all there in Windows 95. Throw in pre-emptive multitasking, an integrated object model, and enterprise connectivity…and you've got a very powerful platform for creating a whole new generation of insanely great applications.

You're wondering how to get involved? We're not talkin' about life in the slow lane here. Register for the 10th Annual MacHack today, and learn how your Mac experience will help you join your friends in programming for Windows 95.

We just shipped a Windows 3.1 version of Ray Dream Designer last October and sales have been excellent, rapidly equaling our Mac sales. The market for graphics applications on Windows is wide-open. And because of Windows 95's new development tools and features (such as memory-mapped files, threads, and image color matching), our engineering team is having a lot of fun programming for Windows 95. Come on in; the water's fine!

Eric Hautemont, President, Ray Dream, Inc.

Photo: Andrea Carr-Lopinto

The transition to Windows 95 offers smaller, fast-moving companies the same great opportunities found in the early days of the Macintosh. Windows 95 is where the action is.

Martin Mazner, former Publisher of MacUser Magazine

MacHack

In response to customer demand, MacHack is once again helping its attendees explore their cross-platform development options. This year, MacHack is proud to offer you a two-day, pre-conference seminar on Windows 95 Programming for Macintosh Developers. A two-day version of the one-day seminars you have heard about at MacWorld. The MacHack seminar will go into the depth and detail that the MacHack audience demands. You'll be introduced to Windows 95 by the man who introduced you to the Macintosh—Stephen Chernicoff, author of the seminal *Macintosh Revealed* series. This seminar will show you how to get the most out of:

The Win32 API • Virtual Memory Management
Memory-Mapped Files • Processes and Threads
Structured Exception Handling
Windows 95's User Interface Architecture
Development Tools • Code Portability

This MacHack seminar won't turn you into a Windows programming guru overnight, but it will get you off to a running start. And because it's right before MacHack, and in the same location, you can easily attend MacHack as well. The seminar will start on the morning of Tue. June 20, and end the evening of Wed. June 21—right before MacHack kicks off with its keynote address late Wednesday night.

Windows 95 Seminar: Tues., June 20—Wed., June 21
$199 if you pre-register before April 16; $249 thereafter
MacHack '95: Thurs., June 22—Sat., June 24
$375 if you pre-register before April 16; $475 thereafter
Location: Ramada Inn, Southfield, Michigan (fly into Detroit)

**To register or for more details,
call Expotech at (313) 882-1824,
or send email to EXPOTECH on AppleLink or
expotech@aol.com on Internet.**

Don't WaitNextEvent—HUnlock the Gestalt of Windows 95 today!

PROGRAMMER'S CHALLENGE

By Mike Scanlin, Mountain View, CA

HY-PHEN-A-TION

Hyphenation algorithms come in two flavors: rule-based and dictionary-based. Of the two, dictionary-based is more reliable but has the downside of requiring a lot of storage space. Rule-based methods require considerably less space and have the option of using an "exceptions" dictionary to improve accuracy. This month's *Challenge* is to implement a rule-based hyphenation algorithm.

The algorithm is this: Each part of the word on either side of the hyphen must include a vowel, not counting a final e es or ed. The part of the word after the hyphen cannot begin with a vowel or a double consonant. No break is made between any two of the following letter combinations: sh, gh, ph, ch, th, wh, gr, pr, cr, tr, wr, br, fr, dr, vowel-r, vowel-n, or om. That means that if any two of those pairs occur next to each other, you can't break them apart (i.e. 'from' contains 'fr' and 'om', which are both on the list; therefore you would not split it between the 'r' and the 'o'). The letter 'y' is not a vowel.

The two routines you'll write are:

```
void *
InitHyphenation(maxRAM)
ulong    maxRAM;

void
Hyphenate(privateDataPtr, inPtr, outPtr)
void   *privateDataPtr;
Str255 *inPtr;
Str255 *outPtr;
```

The `InitHyphenation` routine is an untimed routine called once that can set up whatever tables you might want to use. It must not allocate more than maxRAM bytes, which will be between 16K and 64K.

`Hyphenate` is the routine that does the work. `PrivateDataPtr` is the return value from the InitHyphenation routine. `InPtr` is a pointer to a read-only unhyphenated Pascal word (containing only letters 'a'..'z' and 'A'..'Z'). `OutPtr` is a pointer to 256 bytes of space where you store the hyphenated word. You need to preserve the case of the input and you should insert a `kHyphen` byte (0x2D) everywhere the above algorithm tells you to (yes, the complete output is guaranteed to fit within a Str255).

Note that this algorithm is valid for English only. It will find about 70% of all valid hyphens and will make mistakes about 45% of the time. It is assumed that if this was part of a real application that a hyphenation exception word list would be kept. That list would be checked before calling this function. You, however, don't need to be concerned with that.

Write to me if you have any ques-tions.

TWO MONTHS AGO WINNER

Congrats to **Gustav Larsson** (Mountain View, CA) for winning the Method Dispatcher Challenge. Gustav was a virtual unknown to this column a few months ago but he is rising fast in the Top 20 rankings. The 20 points he wins this month move him from 10th place to 4th place.

Here are the times and code sizes for each entry. Numbers in parens after a person's name indicate that person's cumulative point total for all previous Programmer Challenges, not including this one:

Name	time	code
Gustav Larsson (40)	261	1040
Kevin Cutts (36)	304	266
Jeff Mallett (27)	338	1192
Xan Gregg	349	158
Ernst Munter (51)	385	1466
Thomas Studer	437	572
David Howarth	632	130
Scanlin's brute force method	420	122

Everyone who entered implemented some kind of cache. Gustav chose a 10-way set-associative cache. He splits the 16K of usable cache space into 256 cache sets, each of which holds 10 entries.

THE RULES

Here's how it works: Each month we present a new programming challenge here. First, write some code that solves the challenge. Second, optimize your code (a lot). Then, submit your solution to MacTech Magazine. We choose a winner based on code correctness, speed, size and elegance (in that order of importance) as well as the postmark of the answer. In the event of multiple equally-desirable solutions, we'll choose one winner at random (with honorable mention, but no prize, given to the runners up). The prize for each month's best solution is $50 and a limited-edition "The Winner! MacTech Magazine Programming Challenge" T-shirt (not available in stores).

To help us make fair comparisons, all solutions must be in ANSI compatible C (e.g. don't use Think's Object extensions). Use only pure C code. We disqualify any entries with any assembly in them (except for challenges specifically stated to be in assembly). You may call any Mac routine (e.g., it doesn't matter if you use NewPtr instead of malloc). We test entries with the FPU and 68020 flags turned *off* in THINK C. We time routines with the latest THINK C (with "ANSI Settings", "Honor 'register' first", and "Use Global Optimizer" turned on), so beware if you optimize for a different C compiler. **Limit your code to 60 characters wide**; this helps with e-mail gateways and page layout.

We publish the solution and winners for this month's Programmers' Challenge two months later. All submissions must be ***received by*** the 10th day of the month printed on the front of this issue.

You can get a headstart on the challenge by reading the online version. We post it to the online services at the same time that we post source code. We'll make every effort to have it online no later than when the magazines get mailed out, but we're unable to guarantee that it will be online by any given date.

Mark solutions "Attn: Programmers' Challenge Solution" and send them by e-mail – *Internet* progchallenge@xplain.com, *eWorld* MT.PrgChal, *AppleLink* MT.PROGCHAL, *CompuServe* 71552,174 and *America Online* MT PRGCHAL. Include the solution, all related files, and your contact info. For more details, see "How to Contact Us" on p. 2.

MacTech Magazine reserves the right to publish any solution entered in the Programming Challenge of the Month. Authors grant MacTech Magazine the non-exclusive right to publish entries without limitation upon submission of each entry. Authors retain copyrights for the code.

FREE INSTALLER!

To: Mac Developers & Product Managers

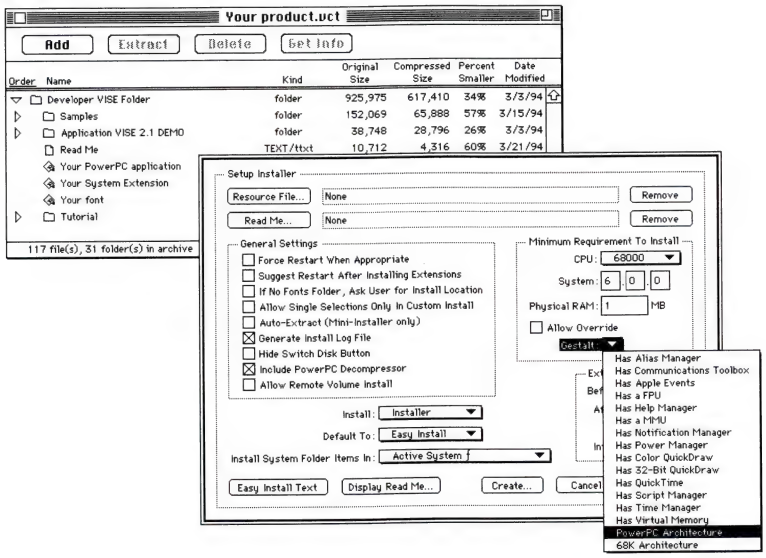
From: MindVision (Creators of Stacker for Macintosh)

Subject: Free Copy of DEVELOPER VISE 3.0

Message: We've got a great new installer for you.
Here's your chance to try it for free.
No risk, no obligation, no hassle.
Call today, don't delay!

Full PowerPC Support

Integrated compression



No programming required

Fully graphical interface

Call (402) 477-3269

IF YOU PREFER, USE E-MAIL. APPLELINK, AOL: MINDVISION • COMPUSERVE 70253,1437

Join the growing list of companies who use our VISE technology: Adobe, Apple, CE Software, Claris, CompuServe, DeltaPoint, MacroMedia, Radius, Stac Electronics, Symantec, WordPerfect, and many more.

MindVision Software

MindVision Software 840 South 30th St., Suite C, Lincoln, Nebraska 68510
Voice: (402) 477-3269 Fax: (402) 477-1395 AppleLink, AOL: MindVision

VISE COMPRESSION TECHNOLOGY

He uses a hash based on the class and method numbers to map to one of the 256 caches. Once he's got that he checks the 10 entries for a match. If he doesn't find a match he uses an efficient binary search to look for the method within the class.

In designing any cache, the choice of hash function and amount of set-associativity (if any) is highly dependent on your data and access pattern. Gustav's code could be tuned for a variety of efficient cache uses. Nice job, Gustav.

Poker Winner Disqualified

Turns out that I failed to do adequate testing on the winning entry for the Poker Hand Evaluator Challenge, in which **Kevin Cutts** was the published winner. Unfortunately, I'm going to have to retro-actively disqualify Kevin and award the 1st place prize to **Gustav Larsson**. My apologies to Kevin and the other MacTech readers. Rather than publish Gustav's well-commented winning solution (it's quite long) I'll make it available via e-mail. If you're interested, send me a note at scanlin@genmagic.com or at any of the Programmer Challenge electronic addresses (see p. 2).

This latent win for Gustav means that he's 3 for 3 in the last 3 challenges, which is an unmatched winning streak. Congrats, Gustav! Also, **Dave Darrah** was the first person to point out the flaws in Kevin's code and so he receives 5 extra points in the cumulative point totals for doing so. Thanks, Dave!

Top 20 Contestants of All Time

Here are the Top 20 Contestants for the 33 Programmer's Challenges to date. The numbers below include points awarded for this months' top 5 entrants. (Note: ties are listed alphabetically by last name -- there are 23 people listed this month because 7 people have 20 points each.)

1.	Boonstra, Bob	176
2.	Karsh, Bill	71
3.	Stenger, Allen	65
4.	Larsson, Gustav	60
5.	Munter, Ernst	53
6.	Riha, Stepan	51
7.	Goebel, James	49
8.	Cutts, Kevin	46
9.	Nepsund, Ronald	40
10.	Vineyard, Jeremy	40
11.	Darrah, Dave	34
12.	Mallet, Jeff	34
13.	Landry, Larry	29
14.	Elwertowski, Tom	24
15.	Kasparian, Raffi	24
16.	Lee, Johnny	22
17.	Anderson, Troy	20
18.	Burgoyne, Nick	20
19.	Galway, Will	20
20.	Israelson, Steve	20
21.	Landweber, Greg	20
22.	Noll, Bob	20
23.	Pinkerton, Tom	20

There are three ways to earn points: (1) by scoring in the top 5 of any challenge, (2) by being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a challenge that I use. The points you can win are:

1st place . . . . . .20 points
2nd place . . . . . .10 points
3rd place . . . . . .7 points
4th place . . . . . .4 points
5th place . . . . . .2 points
finding bug . . . . . .5 points
suggesting challenge . . . . .2 points

Here is Gustav's winning solution:

DISPATCH.C


```
#define kMethodNotFound ((void *) 0)
#define kClassNotFound  ((void *) -1)

typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned long ulong;

typedef ushort  ClassID;
typedef ushort  MethodNumber;
typedef void *  MethodAddress;

typedef struct {
  MethodNumber  methodNumber;
  MethodAddress methodAddress;
} MethodEntry;

typedef struct {
  ushort        inheritedCount;
  ushort        inheritedClasses[15];
  MethodNumber  largestMethodNumber;
  ushort        methodCount;
  MethodEntry   methods[];
} Class, *ClassPtr;
```

/* Each block in the cache is 64 bytes and holds 10 entries. The id field distinguishes class/method pairs that map to the same cache block. The hits field has a bit for each entry, which is set whenever there is a hit on the entry. The "next" field is 0..9 and indicates where to check first when deciding which entry to replace. */

```
#define BLOCK_SIZE 10

typedef struct {
  ushort          id [ BLOCK_SIZE ];
  ushort          hits;
  ushort          next;
  MethodAddress   methodAddress [ BLOCK_SIZE ];
} CacheBlock;

static CacheBlock Cache[256]; /* exactly 16K */
```

GetClassPtr

```
extern ClassPtr GetClassPtr( ClassID );
MethodAddress FindMethod( ClassID, MethodNumber );
static MethodAddress HandleMiss( ClassID, MethodNumber,
                                 CacheBlock *, ushort );
```

FindMethod

```
MethodAddress
FindMethod( ClassID classID, MethodNumber methodNumber )
{
  register ulong hash;
  register ushort *idPtr;
  register CacheBlock *block;
```

Check cache for this class & method.

/* First, generate a hash key that is unique for each class/method pair. The formula classID + 437 * method works since 437 is greater than the highest class ID, 400. The ideal multiplier depends heavily on the distribution of class/method pairs, but 437 seems to give a reasonable spread under a variety of conditions.

The low byte of the hash key becomes the block number, and the higher bytes become the "id" (to distinguish class/method pairs that map to the same block). Add one to the id so it is never zero (zero indicates an unused cache entry). */

```
  hash = classID + 437L * methodNumber;
  block = &Cache[ (uchar)hash ];
  hash = (hash >> 8) + 1;    /* now hash holds just the id */
```

WWWeapon of Choice.

Wield BBEdit. Become a net.warrior.

BBEdit 3.1 has powerful features for the Internet information provider as well as the 'net surfer.

- Integration with Internet Config (included on the CD) for access to "helper applications" such as Web browsers, e-mail clients and FTP clients.
- HTML formatting extensions for the production of World Wide Web browser documents.
- Automatic line feed translation for easy interchange with DOS and Unix file formats.
- Fast and flexible text searching, including multi-file search and optional 'grep' matching.

BBEdit 3.1 has the power and versatility.

The features that you have come to count on have been expanded and improved. Here are just a few:

- PowerPC™ code to speed editing, searching and transformation of text.
- Macintosh® Drag and Drop for editing of text.
- Scriptable with AppleScript, Frontier 3.0 or any other OSA-compatible scripting architecture.
- Automatic "soft" text wrapping.
- The integrated PopupFuncs™ technology recognizes C, C++, Pascal, Rez, Fortran, and 68k assembler for speedy navigation of source code.
- Unique "Find Differences" command to interactively compare files, folders or projects.
- Integrated support for Metrowerks CodeWarrior, Symantec C++ (including the new version 8.0), Think C, THINK Reference, and MPW ToolServer.

Grab some goodies and try out some cool software.

With 600 megabytes of room to spare we are able to give you more Bare Bones Software for your money. Groove on these goodies:

- BBEdit Extension Developer's Kit.
- BBEdit freeware and shareware extensions from developers around the world.
- Tools from Bare Bones Software's freeware collection, including Drop•PS and Convert• Projects.

The CD also includes demo versions of many popular development tools, updates, information and special offers from industry leaders. Have a seat and check out the stuff from these guys:

- Aladdin Systems
- Language Systems
- MacTech Magazine
- Mathemæsthetics
- MindVision Software
- Onyx Technology
- Peirce Software
- Symantec

"How do I get mine?", you ask?

During the World Wide Developers Conference you can purchase BBEdit 3.1 from APDA or MacTech at their respective booths. BBEdit is also available in the Mail Order Store section of this issue of MacTech or direct from Bare Bones Software.

BBEdit... It doesn't suck.

Bare Bones Software, Inc.

For a free demo disk, send us e-mail!

Internet: bbsw@netcom.com
CompuServe: 73051,3255
Applelink: BARE.BONES
eWorld: BareBones

BMUG
SPRING 1995
CHOICE PRODUCT

Accelerated for Power Macintosh

Mac™ OS

```
  idPtr = block->id;
  if ( *idPtr++ == (ushort)hash ) goto hit0;
  if ( *idPtr++ == (ushort)hash ) goto hit1;
  if ( *idPtr++ == (ushort)hash ) goto hit2;
  if ( *idPtr++ == (ushort)hash ) goto hit3;
  if ( *idPtr++ == (ushort)hash ) goto hit4;
  if ( *idPtr++ == (ushort)hash ) goto hit5;
  if ( *idPtr++ == (ushort)hash ) goto hit6;
  if ( *idPtr++ == (ushort)hash ) goto hit7;
  if ( *idPtr++ == (ushort)hash ) goto hit8;
  if ( *idPtr++ == (ushort)hash ) goto hit9;

  return HandleMiss( classID, methodNumber,
                     block, (ushort)hash );
```

/* Handle cache hit */

```
hit0: block->hits |= 0x001;  return block->methodAddress[0];
hit1: block->hits |= 0x002;  return block->methodAddress[1];
hit2: block->hits |= 0x004;  return block->methodAddress[2];
hit3: block->hits |= 0x008;  return block->methodAddress[3];
hit4: block->hits |= 0x010;  return block->methodAddress[4];
hit5: block->hits |= 0x020;  return block->methodAddress[5];
hit6: block->hits |= 0x040;  return block->methodAddress[6];
hit7: block->hits |= 0x080;  return block->methodAddress[7];
hit8: block->hits |= 0x100;  return block->methodAddress[8];
hit9: block->hits |= 0x200;  return block->methodAddress[9];
}
```

HandleMiss

```
static MethodAddress
HandleMiss( ClassID classID, MethodNumber methodNumber,
            CacheBlock *blockPtr, ushort id )
{
  register ClassPtr classPtr;
  MethodAddress methodAddress;
  register ushort *temp;   /* shared address register */

  /* Not in cache, so look it up the hard way */
  classPtr = GetClassPtr( classID );
  if ( classPtr != kClassNotFound ) {

    /* Look in this class. Use a binary search. */
    {
      register MethodEntry *methods = classPtr->methods;
      ushort searchSize = classPtr->methodCount;
      register MethodNumber methodReg = methodNumber;
                /* method # in a register */
```

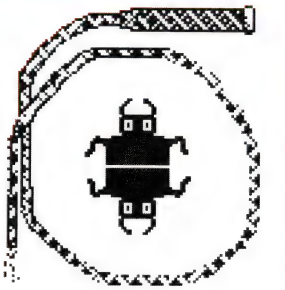
/* Unroll the binary search for the most common cases. The BINARY macro reduces the search to progressively more elementary cases. If classPtr->methodCount is greater than 32, we run a general binary search until the search is reduced to an unrolled case. */

```
continueBinarySearch:
      switch ( searchSize )
      {
        bin2: case 2:
          if ( methods[1].methodNumber == methodReg ) {
            methodAddress = methods[1].methodAddress;
            goto addCache;
          }
    /* else fall through... */
        bin1: case 1:
          if ( methods[0].methodNumber == methodReg ) {
            methodAddress = methods[0].methodAddress;
            goto addCache;
          }
          else goto checkSuperclasses;

      #define BINARY(mid,mid1)                          \
        if ( methodReg < methods[mid].methodNumber )    \
          goto bin##mid;  /* like "hi = mid" */          \
        else {                                           \
          methods += mid; /* like "lo = mid" */          \
          goto bin##mid1;                                \
        }

        /* binN: case N: BINARY(N/2,(N+1)/2) */
        bin3:  case  3: BINARY(1,2)
        bin4:  case  4: BINARY(2,2)
        bin5:  case  5: BINARY(2,3)
        bin6:  case  6: BINARY(3,3)
        bin7:  case  7: BINARY(3,4)
```

Gives you the Information to *Program your Best!*

The Debugger V2 & MacNosy

by Steve Jasik



Information

Control

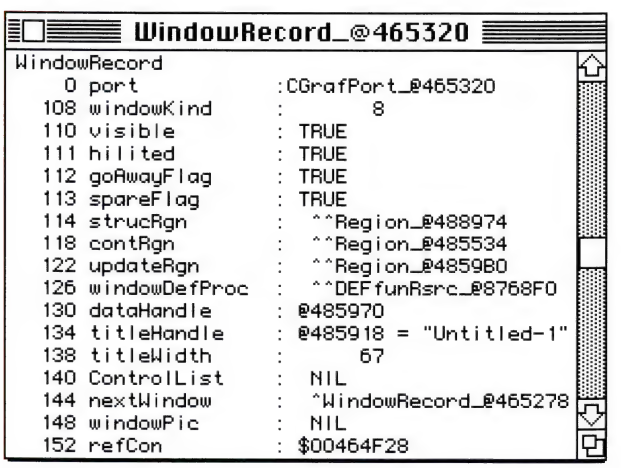
The Debugger is a low and high-level symbolic Debugger that runs in a full multi-window Macintosh environment. You can trace program execution, view the values of variables, etc. ***of both 68K and PowerPC programs***.

MacNosy is a global interactive disassembler that enables one to recover the source code of any Mac application, resource file or the ROM.

When you compare features of the different debuggers, note that *only one* has all the below features to help you get your job done, and *only one* has MacNosy to help you debug any program in a full system (6.0x or System 7.x) environment symbolically!

It is the *only* debugger to use the MMU to protect your CODE resources and the rest of the system from the program you are debugging. With MMU Protection you can find errors when they happen, not millions of instructions later! (Macintoshs with 68030 CPUs only).

The Debugger is the debugger of choice at: Adobe, Aldus, Claris, Electronic Arts, Kodak, Metrowerks, etc.



An example of a structured data display window

Its Features Include:

- **Symbolic Debugging** of *any* Macintosh program, ROM, or *code resource* (DRVRs, XCMDs, INITs, PDEFs, 4DEXs ..)
- **Source level debugging for Metrowerks & MPW** compiled programs (C++, C, Pascal, Fortran, ...), and an Incremental Build System with instant Link for superfast development.
- **Object Inspector for MacApp 3 programs**
- **Source level debugging of Think C™ projects**
- **Includes a program (CoverTest) to interactively do Code Coverage analysis for SQA testing, etc.**
- **Simultaneous Symbolic debugging of multiple "tasks"**
- **Fast Software Watchpoint** command to find clobbered variables
- **Sophisticated error check algorithms** such as Trap Discipline (Argument Checking), Handle Zapping, Heap Scramble and Heap Check to detect program errors before they become disasters
- **Structured display** of data (hypertext) with user definable structures while debugging
- **Conditional breakpoints** to help filter out redundant information
- **Continuous Animated Step Mode** to watch your program execute instruction by instruction
- **Detailed symbolic disassembly for both 680x0 and PowerPC** with symbol names, labels, cross ref maps, – make it possible to ferret out the secrets of the ROM, etc.
- **"Training Wheels"** for the PowerPC disassembler to help you learn the opcodes

The Debugger V2 & MacNosy: $350
Runs on all Macs. Call For Group prices or Updates. Visa/MC Accepted.

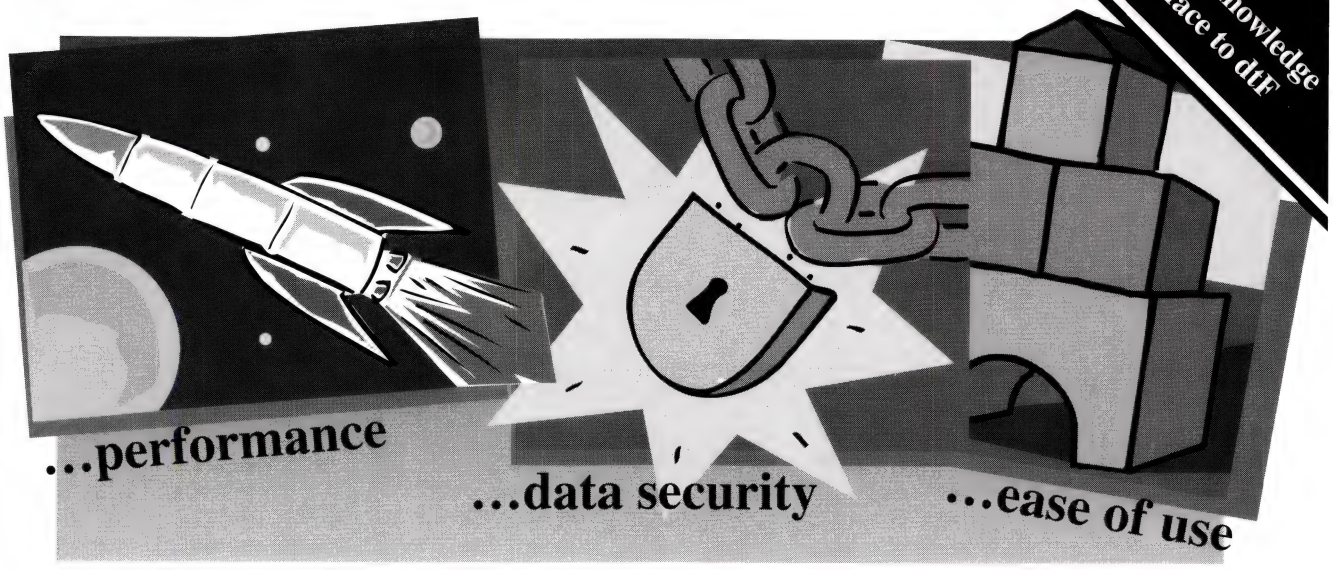
Available from: Jasik, APDA, Frameworks or ComputerWare (800-326-0092).

Jasik Designs • 343 Trenton Way, Menlo Park, California 94025 • **(415) 322-1386**
Internet: macnosy@jasik.com • Applelink: D1037

```
bin8:  case  8: BINARY(4,4)
bin9:  case  9: BINARY(4,5)
bin10: case 10: BINARY(5,5)
bin11: case 11: BINARY(5,6)
bin12: case 12: BINARY(6,6)
bin13: case 13: BINARY(6,7)
bin14: case 14: BINARY(7,7)
bin15: case 15: BINARY(7,8)
bin16: case 16: BINARY(8,8)
bin17: case 17: BINARY(8,9)
bin18: case 18: BINARY(9,9)
bin19: case 19: BINARY(9,10)
bin20: case 20: BINARY(10,10)
bin21: case 21: BINARY(10,11)
bin22: case 22: BINARY(11,11)
bin23: case 23: BINARY(11,12)
bin24: case 24: BINARY(12,12)
bin25: case 25: BINARY(12,13)
bin26: case 26: BINARY(13,13)
bin27: case 27: BINARY(13,14)
bin28: case 28: BINARY(14,14)
bin29: case 29: BINARY(14,15)
bin30: case 30: BINARY(15,15)
bin31: case 31: BINARY(15,16)
bin32: case 32: BINARY(16,16)

#define NUM_UNROLLED 32 /* # of unrolled cases */

default:
  if (methodReg <= classPtr->largestMethodNumber) {
    register ushort lo, mid, hi, mn;
    lo = 0;
    hi = classPtr->methodCount - 1;
    while ( lo + NUM_UNROLLED - 1 < hi ) {
      mid = (lo + hi) >> 1;
      mn = methods[mid].methodNumber;
      if ( methodReg < mn )
        hi = mid;
      else if ( methodReg > mn )
        lo = mid;
      else {
        methodAddress = methods[mid].methodAddress;
        goto addCache;
      }
    }
    /* reduced to an unrolled case */
    searchSize = hi-lo+1;
    methods += lo;
    goto continueBinarySearch;
  }
 }
}

/* Look in superclasses. Eliminating the recursion by keeping a custom stack of critical variables doesn't buy us much. Recursion also makes the code clearer. */

checkSuperclasses:
 {
  register ulong i, max = classPtr->inheritedCount;
  /* shared address register points into class list */
  temp = classPtr->inheritedClasses;

  for ( i = 0; i < max; i++ ) {
    methodAddress = FindMethod(*temp++, methodNumber);
    if ( methodAddress != kMethodNotFound )
      goto addCache;
  }
 }
}

/* There are two ways to get here: classPtr is kClassNotFound (hopefully rare), or the method was not found in any of the superclasses. Either way, put a "not found" entry into the cache. This helps when searching complicated inheritance hierarchies or repeatedly looking up the same method in several related subclasses. */

  methodAddress = kMethodNotFound;

---

addCache:                                   Add an entry to the cache.

// shared address register holds block pointer
  temp = (ushort*) blockPtr;
  #define block ((CacheBlock*) temp)
  {
    register ulong hits = block->hits;
    register ulong next = block->next;
    register ulong mask = 1L << next;

/* Choose the entry to replace. Look for a cleared bit in "hits" starting at "next". If all the bits are 1 initially, go all the way around; we are guaranteed to stop since we clear bits as we go. */

    while ( hits & mask ) {
      hits &= ~mask;                    /* no, clear the bit */
      mask <<= 1;                       /* and try next bit */
      if ( mask == (1 << BLOCK_SIZE) )
        mask = 1;
      next++;
    }
    if ( next >= BLOCK_SIZE )
      next -= BLOCK_SIZE;

    block->methodAddress[ next ] = methodAddress;
    block->id[ next ] = id;
    block->hits = hits;
    block->next = ( next < BLOCK_SIZE-1 ? next+1 : 0 );
  }

  return methodAddress;
}
```

dtF The Relational Database System

New!
DataScript™ by General Knowledge
The AppleScript-Interface to dtF



When it comes to performance **dtF** is in a class all its own. **dtF** utilizes a proprietary query optimization and caching scheme to obtain unparalleled performance. And if you are in need of a quantum leap in performance, use the **dtF** native PowerPC standalone or server version.

Full transaction control and error recovery guarantee maximum data protection even after sudden system crashes. **dtF** databases are compressed and encrypted to protect against all unauthorized access, even disk editors.

Integrated data dictionary, security, automatic index selection, query optimization, deadlock detection and error recovery allow you to concentrate on your application. The **dtF** API was designed for a seamless integration with third-party toolkits and class libraries.

dtF features include SQL, BLOB support, electronic documentation, sample applications, tutorials, and tools for importing, exporting, creating, managing databases and users. **dtF** is currently available for Macintosh, DOS, Windows, OS/2. The C/C++ API is identical across all platforms.

Available for Macintosh System 7.x and native PowerPC. MPW C/C++ (68K and native PPC), Symantec C/C++, Metrowerks CodeWarrior (68K and native PPC). Separate versions for use with HyperCard, SuperCard, SmalltalkAgents® and Prograph CPX available.

Standalone applications built with dtF are royalty-free.

dtF Americas, Inc.
19672 Stevens Creek Blvd.,
Suite 128
Cupertino, CA 95014, USA

Phone: (800) DTF-1790
Fax: (510) 828-8755
AppleLink: DTF.AMERICA

dtF The Relational Database System

SYMANTEC TOP 10

By Colen Garoutte-Carson, Quality Assurance & Technical Support, Symantec Corp.

SYMANTEC.™

This monthly column, written by Symantec's Technical Support Engineers, aims to provide you with technical information based on the use of Symantec products.

Q: How do I build a fat application with Symantec C++ v8.0?

A: We've touched on this before, and here's a more detailed look. When an application is launched on a 68K Macintosh, the operating system (OS) loads and executes a 68K `'CODE'` resource ID 1. When an application is launched on a PowerMac, the OS first looks for a `'cfrg'` resource describing a PowerPC code fragment stored in the application's data fork, and if found, prepares and executes the PowerPC code fragment. If the OS is unable to launch the application this way, it tries to loads and execute a 68K `'CODE'` resource ID 1.

A fat (fat-binary) application has both a PowerPC code fragment and 68K `'CODE'` resources, and can be launched on either platform.

To create a fat binary app, enable the option to merge an existing 68Kx0 application into your final application, in the project **Options...** dialog, in the **Project Type** settings for an application. This copies all resources from the 68K application that do not already exist in your PowerPC application. Most importantly, this copies all 68K `'CODE'` resources to your PowerPC application.

Resources in your 68K application with the same type and ID as those in your PowerPC application will not be copied.

You can also build a fat application by including a 68K application's `'CODE'` resources in a resource (`.rsrc`) file, and adding that resource file to your PowerPC project.

Q: How do I create an accelerated code resource?

A: Although the linker in Symantec C++ v8.0 does not yet support the creation of code resources, there is a way to create a code resource from a PowerPC application using Symantec Rez. Symantec C++ v8.0 includes a Code Resource project model which does this.

The Code Resource project model consists of two projects. The first, `Code Resource.π`, is similar to a `Mac Application` project, but doesn't include `InitToolbox()` (it's unnecessary in a code resource). It uses the `rsrcMPWPPCRuntime.o` runtime library instead of `PPCRuntime.o`, and `PPCLink & MakePEF` instead of the internal linker.

In PowerMac applications, constructors and destructors for global and static objects are called automatically by `__cplusstart()`, which the linker assigns as the main entry point to your application (`not main()`). This routine calls your constructors, initializes your QuickDraw globals, calls your `main()` function, and then calls your destructors.

In a shared library, constructors for global and static objects are called by `__cplusinit`, which the linker assigns as an initialization routine. The Code Fragment Manager will automatically call this initialization routine when your shared library is loaded. The `__cplusinit()` routine will also hook your shared library's QuickDraw globals into the QuickDraw globals of the calling application. Destructors for global and static objects are called by `__cplusterm()`, which the linker assigns as a termination routine. The Code Fragment Manager will automatically call this termination routine when your shared library is unloaded.

In code resources, there is no way to automatically call constructors or destructors for global and static objects, or hook QuickDraw globals, and still preserve the arguments passed to `main()`. Instead, you must call routines to do this for you. In C, you use `__rsrcinit()` and `__rsrcterm()`, declared in `stdlib.h`. In C++, you use `__cplusrsrcinit()` and `__cplusrsrcterm()`, declared in `new.h`. The C++ initialization and termination routines call the C initialization and termination routines. Use only the C++ or C initialization routines. Do not use both. These routines are defined in the `rsrcMPWPPCRuntime.o` library.

If your code resource is called only once to perform its function, call `__rsrcinit()` or `__cplusrsrcinit()` at

Cross-Platform Object Database Engine

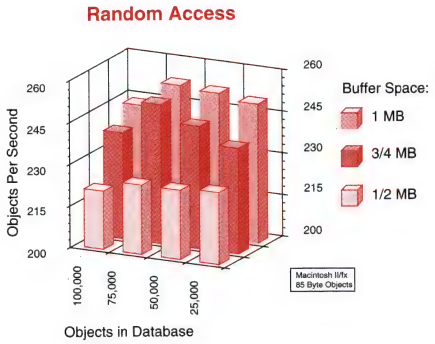
neoAccess™

Full Featured

NeoAccess™ allows you to embed the power of a full-featured object-oriented database engine into your **Macintosh**, **Windows** and **DOS** based applications. No other object database engine can match NeoAccess's impressive feature set, including: Blobs, part lists, iterators, swizzlers, temporary objects, multiple indices on a class, schema evolution, a powerful relational query mechanism, a streams-based i/o model and incredible performance.

High Performance

Internally, NeoAccess uses extended binary trees and binary search algorithms to achieve optimally short access times. Its automatic query optimizer ensures that queries always use the fastest access path to objects. And indices are dynamically combined, collapsed and compressed to keep access times to an absolute minimum as the contents of a database changes. NeoAccess's object caching boosts performance by keeping objects in memory even after being disposed of by the application. Your application's memory size is reduced because only those objects of immediate interest need to be in memory at any one time, not the entire file.



Full Source Code

We've taken a frameworks approach toward object persistence and database technology. In much the same way that application frameworks are used to construct the front-end of an application, NeoAccess is the framework you use to build your application's back-end. As is the case with virtually all object frameworks, the NeoAccess Developer's Toolkit comes complete with ***full source code***, for all major application frameworks including Metrowerks' **PowerPlant**, Symantec's **THINK Class Library** and Apple's **MacApp** on the Macintosh and **Microsoft's Foundation Classes**, Inmark's **zApp** and Borland's **ObjectWindows** in Intel-based environments. It can even be used without a framework or in one that you've designed.

Easy to Use

The programming interface is designed around the concept of minimum visible complexity. Application-specific objects inherit persistence properties from a NeoAccess base class. These objects are organized in the database primarily by class. But NeoAccess also knows how classes are related. So multiple classes can be searched in a single operation. And of course objects of any particular class can be organized using multiple indices. NeoAccess is unique in that it allows objects to be located based on abstract selection criteria or based on their relationship to other objects. There's literally no database administation to deal with – NeoAccess takes care of all the details. NeoAccess also includes a Blob mechanism which allows free-form variable-length data to be stored in databases with the same ease as fixed-length objects. NeoAccess even includes a powerful set of keyed iterators for traversing indices and part lists. Keyed iterators have the unique ability to iterate over only those objects in a set that match a given selection criteria. Your users will appreciate NeoAccess because databases are completely self-contained in a single document file. So users can treat a database file as they would any other document.

Proven

NeoAccess has been commercially available since May of 1992. Hundreds of commercial and in-house applications based on NeoAccess technology have already been deployed. NeoAccess can help your organization deliver powerful products in a more timely fashion than you ever imagined possible.

Affordable

NeoAccess's best feature is its price. The NeoAccess Developer's Toolkit sells for just $749 per developer with absolutely ***no runtime licensing fees***. It includes ***full source code***, numerous sample applications, 450+ pages of documentation, and 30 days of technical support. So what are you waiting for?

Now on the CodeWarrior CD!
See the NeoLogic folder on the root of the CD-ROM for more details.

neo•logic

In order to survive, you need to persist.

No Runtime Licensing Fees!
Use NeoAccess in all your development for one low price!

1450 Fourth St. • Suite 12 • Berkeley • CA • 94710 • (510) 524-5897 • neologic@mail.dnai.com

the start of your program, and `__rsrcterm()` or `__cplusrsrcterm()` at the end.

If your code resource is reentrant, call `__rsrcinit()` or `__cplusrsrcinit()` only the first time your code resource is called, and `__rsrcterm()` or `__cplusrsrcterm()` the last time your code resource is called.

Do not use any QuickDraw globals, or any global or static objects that rely on their constructor, before calling `__rsrcinit()` or `__cplusrsrcinit()`.

The second project in the Code Resource project model will copy the data fork of the target built by `Code Resource.π` into a resource and prepend a routine descriptor to it. `Code Resource.π` is included as a sub-project to the second project. Debugging is disabled for the sub-project to ensure that the sub-project's target is built instead of a temporary instant-run PEF when the parent project is brought up to date.

`Code Resource.π` includes two resource (`.r`) files, `_CopyToResource.r` and `__AppendRoutineDescriptor.r`.

`_CopyToResource.r` reads the data fork of the sub-project's target and stores it into a temporary resource.

`__AppendRoutineDescriptor.r` prepends a routine descriptor to the temporary resource and stores it in your final resource. Before you build your code resource, you will need to modify a number of `#define`'s in this file. The file contains comments that describe the necessary changes.

The **Always check file dates** option is enabled in the second project to ensure that the resource description files' dependency on the sub-project's target is properly tracked.

`__AppendRoutineDescriptor.r` is dependant on `_CopyToResource.r`. These files have been named to compile in the right order.

Once the parent project has been brought up to date, your accelerated code resource will be stored in the project's resource file (`<projectname>.rsrc`).

Q: How do I create a fat or safe fat code resource?

A: You can build a fat or safe fat code resource in much the same way that you would create an accelerated code resource, discussed above. The `Code Resource` project model includes support for these fat code resource types, as well as a few others. See the comments within the project model for details.

Q: Why are my structs a different size in my PowerPC project than they are in my 68K project?

A: The MC68000 chip must reference word or long values from even addresses. More recent chips in the 680x0 family do not have this limitation, but perform faster when referencing word or long values from even addresses. The 68040 and PowerPC chips perform even faster with structures aligned on 4-byte boundaries.

If you need to share data structures in memory between 68K and PowerPC code (e.g. when passing data to the Macintosh Toolbox's pre-existing 68K-based interfaces), or need

Let's Table the discussion.

Take data from multiple sources and create an **unlimited** number of rows and columns made up of text, movies, pictures, ICONs, SICNs and other types of information.

AppleScript support provides integratation with other applications. Full set of object resolvers provided.

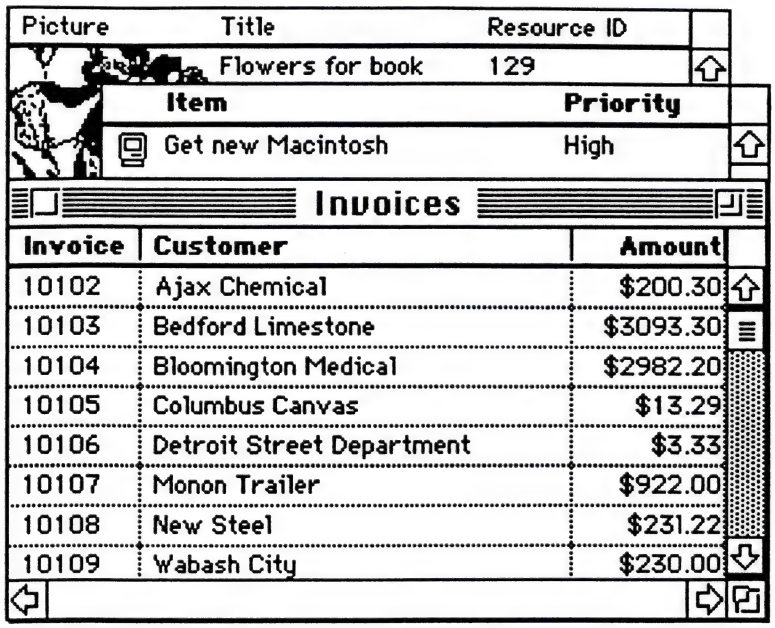
Build powerful tables in your C/C++ environments by replacing the list manager with TableIt!. Our drawing procedures make LDEF's a thing of the past in under 5 minutes.

Leap into the future with support for **OpenDoc™**, the exciting new compound document architecture. Combine TableIt! parts with other commercial and custom parts to provide exciting and powerful solutions.

Enlarge the amount of data you can work with in your application by combining TableIt! with NeoAcess™ - the powerful cross-platform object-database. No more RAM limits!

Integrate TableIt! with your current application environments - 68K and PowerPC versions are available for 4D, HyperCard, SuperCard as well as most C/C++ compilers. *Certified* PowerPC native for CodeWarrior.

Test drive a version on a 30 money back guarantee.



TableIt! Features:

- Unlimited number of rows and columns - go past the typical 32K limitation!
- Support for core, database, table and text suites. Playable, Recordable and Attachable.
- Over 100 properties that provide customized look and feel.
- Drag and Drop Support.
- In-cell editing
- DR/2 for OpenDoc available now!

Pricing:

$150 retail for stand-alone, HyperCard, SuperCard or 4D. Developers Licenses and OpenDoc part licenses available. No Royalties!

Graphical Business Interfaces, Inc.
Voice: 219-253-8623 Fax: 219-253-7158
E-Mail: info@gbi.com
http://www.gbi.com

TableIt!™ Bringing your data into view.™

to read or write data structures from/to disk with both 68K and PowerPC code, you will need to ensure that your alignment settings match in both your 68K and PowerPC projects.

By default, the PowerPC compilers align fields within structs on 4-byte boundaries. By default, the 68K C++ compiler aligns fields within structs on 2-byte boundaries. Think C always aligns fields within structs on 2-byte boundaries.

The PowerPC compiler supports the following `#pragma` directive to enable 68K (2-byte) structure alignment in a given structure declaration :

```
#pragma options align=mac68K
/* Your struct declaration */
#pragma options align=reset
```

Q: My 68K application, created with the Visual Architect in v7.0(.x), works fine. Why does a fat version of the same application not run on a 68K Mac?

A: Because of changes in the TCL, you will need to re-compile the 68K application with the 68K environment and TCL included in version 8, and re-merge it into your fat application.

Q: When I run my program with the debugger, it launches "Power Mac DebugServices". This program remains running invisibly. How can I quit this application when I am done debugging?

A: You can write an AppleScript that contains the following :

```
tell application "Power Mac DebugServices" to quit
```

Never run this script while debugging. To ensure that DebugServices has quit when no longer needed, name the script `ShutDown`, and put it in your (Scripts) folder. The script will be run when you quit the Symantec Project Manager.

Q: Can I use Apple's new Toolbox Assistant with the Symantec Project Manager?

A: Yes. The Symantec Project Manager supports both The THINK Reference and QuickViewer. It is capable of using both simultaneously.

As with the Think Project Manager, the Symantec Project Manager will allow you to reference the Think Reference or QuickViewer directly if an alias to it, named exactly "THINK Reference" or "Primary Doc Server", is located in your "(Tools)" folder. The Symantec Project Manager considers these names synonymous. The Symantec Project Manager will search a second documentation server if it's unsuccessful in finding a match in the first. The second documentation server's alias must also be located in the "(Tools)" folder and must be named "Secondary Doc Server".

Q: When I convert my Think Project Manager project, the Symantec Project Manager preserves the segmentation in groups. Is there a way around this?

A: Yes, hold down the shift key when opening the Think Project Manager project.

Q: How do I **Check In...** files to my SourceServer database for the first time? And how do I **Check Out...** files from a SourceServer database that I've never checked out before?

A: To do these actions, you must send the command to SourceServer manually. The Symantec Project Manager includes a Worksheet window, available under the **Windows** menu, that you can use to send commands to SourceServer or ToolServer.

Example of a CheckIn command :

```
CheckIn -new "filename.cp"
```

Q: Can I use **Check In...** on the Revision menu to check in header files to a SourceServer database?

A: Yes. In the project **Options...** dialog, on the "Extension" settings page, add `.h` to the list of supported filename extensions, but do not specify a translator. You will be able to add files with names ending in `.h` to your project, but they will not be compiled. You can then check these files into your SourceServer database by selecting them in the project window, and then selecting **Check In...** from the **Revision** menu.

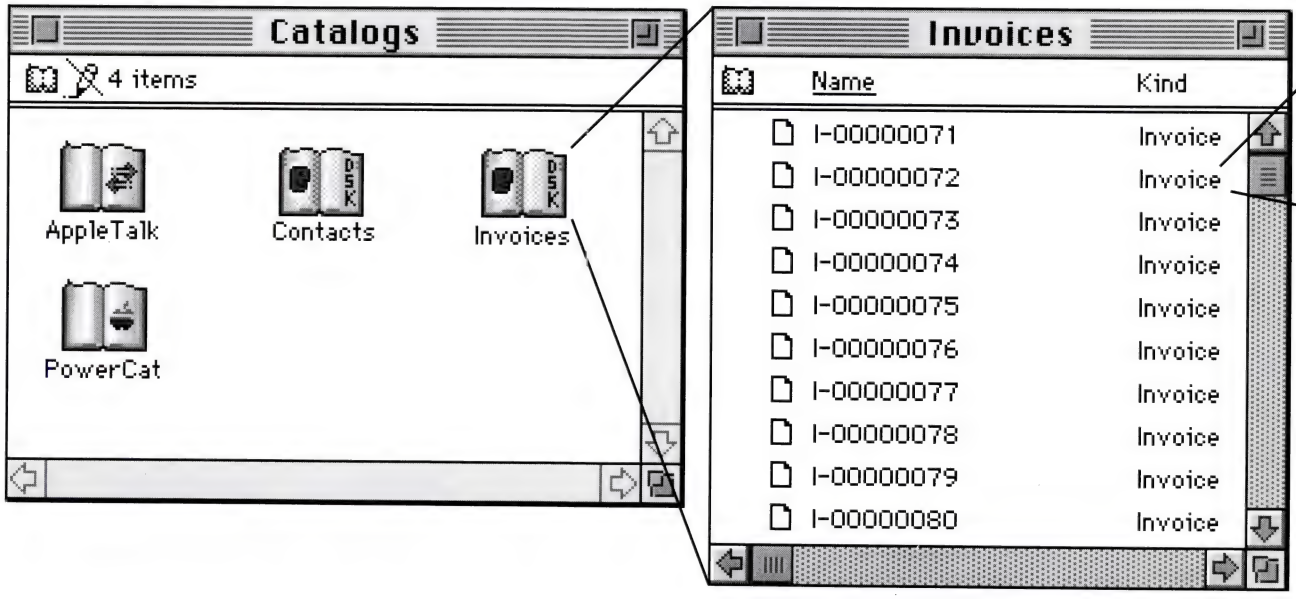
Databases in the Finder?

Unfortunately, most users must interact with relational databases at the SQL level or behind a front-end that can take months to develop.

select invoice_num,invoice_date,cust_id,cust_name from invoices,customers where invoice_id = 112;printall;

Why not view and edit your data at the Finder level? Announcing **Cataloger™**.

Cataloger brings a whole new way of working with databases to the Macintosh. The Finder™ has always provided a view of our desktop files, why not database records too?



Apple's **O**pen **C**ollaborative **E**nvironment places a catalog icon at the Finder. Open the icon up and you'll see a window like the one above. The AppleTalk catalog shows nodes on the network. The PowerCat catalog shows users and groups in a PowerShare Server.

The Contact and Invoices catalogs (for example) can show information from **4D Server**, **DAL/DAM** , **Oracle** and **Sybase** databases! If your needs are more customized, you can write AppleScript handlers to provide the contents of your catalog.

The Catalog Manager (an AOCE API) allows for **C**atalog **S**ervice **A**ccess **M**odules (CSAMs) to be created that interact with external sources.

The Finder and other applications make requests through the Catalog Manager and Cataloger does the rest. There are no limits to the amount of data brought back.

Templates provide the ability to view and edit individual records. AOCE Templates are designed with **Template Constructor™.** This powerful application let's you easily design forms to view data coming from several sources, control behavior with C and AppleScript, and view row/column data with **TableIt!™** - a powerful data display tool.

I-00000072

Invoice *: I-00000072 Invoice Date: 9/1/94

Company: Tall Timbers Marina Terms: Net 30 Days

Qty	Part *	Description	Cost Each	Line Total
1	2143	Propeller	250	250
2	3853	Drain Plugs	12.90	25.80
1	4905	Main Light assembly	150	150
8	9384	Spark Plugs	4.00	32.00
1	9999	Labor	600	600

Notes: Prepared boat for summer season. Dewinterized, tuned up and replaced main propeller. Light assembly replaced after testing front lights failed.

Subtotal: 1057.80

Tax: 52.89

Total: 1110.69

Access information and catalog definitions are stored in and protected by the AOCE **Keychain**. Your users can have one username and password for a variety of data sources!

Need more horsepower? **Database Scripting Kit™** provides AppleScript access to all of the connection information and data sources. Write AppleScript routines that send queries and parse results from within your own application.

Cataloger/Template Constructor includes:

Database Scripting Kit™ with connectors for 4D Server, DAL/DAM, Oracle, Sybase and others.

DSK Cataloger™ CSAM

Template Constructor™ with TableIt!™

Several example catalogs/templates and over 100 example AppleScripts.

Complete scriptability, native for Power Macintosh. Drag Manager and Thread Manager support.

How to contact us:

Graphical Business Interfaces, Inc.

Voice: 800-424-7714 or 219-253-8623

Fax: 219-253-7158

E-Mail: steve@gbi.com

GBI. Bringing your data into view.™

By Kurt Schmucker, Advanced Technology Group, Apple Computer, Inc.

MacApp and Prograph CPX - A Comparison

Designing and implementing an event-driven application with a sophisticated graphical user interface is a difficult task. Fortunately, a number of years ago Larry Tesler and other researchers at Apple came up with a way of making both the design and the implementation of a Macintosh application considerably easier. What they did was to design a working skeleton application that you could easily customize. Unlike earlier skeleton apps, this new one did much more, was more easily extended, and yet required you to learn much less about its inner workings. The main reason they were able to do this was that they used an object oriented language and designed their skeleton as an interlocking set of classes in that language. The resulting skeleton became known generically as an application framework and resulted in an Apple product called "MacApp – The Expandable Macintosh Application". [1] Other application frameworks from other companies followed: some were Macintosh application frameworks and others were for other platforms. However, MacApp, as the first and the generally most full featured, remains the standard against which all new application frameworks are measured.

When CPX first came out, I embarked on a project to compare it to MacApp by reimplementing some of the MacApp sample programs in CPX, seeing what could and couldn't be done, measuring RAM and disk footprints, measuring the performance of the resulting apps, etc. I have gotten far enough on this project to do a fair comparison, and this article will document what I have been able to do, and the conclusions I have reached.

However, not everything in this article will be objective numerical data like RAM footprint measurements. Developers are emotional about their tools, and why shouldn't they be? You spend a great deal of time and effort learning to use a tool, and once mastered, you rely heavily on that tool for your livelihood. I am no exception to this. While I will present the factual data that I was able to generate in doing this comparison, I will also add my own personal opinions and impressions about MacApp and CPX. In addition, since it is difficult to isolate a tool from the environment it's used in, I will also compare the total development experience of MacApp and CPX, including programming languages, build times, auxiliary development tools like direct manipulation window layout editors, etc.

Characteristics of an Application Framework

A Macintosh application framework is a set of classes in some object-oriented language designed in such a way that:

- it is a stand-alone, double-clickable, 'vanilla' Macintosh application. (Because it is an app, it has a working menu bar, can open multiple windows each of which are movable, resizable, etc., works with the clipboard, prints, and exhibits all the other standard behavior that a Macintosh user expects. It is a *vanilla* app since all the windows are empty, the menus are blank, and it prints out blank paper.)
- it can easily be extended by subclassing and overriding. (For example, to get a new type of window that draws three-d renderings of a molecule, you subclass the Window class, and override its Draw method (or the Draw method of an auxiliary object) and include the molecule drawing code in that new method.)
- it provides a set of special null methods that enable the framework to access your code. You override these methods to do routine customization like enabling menu items, processing mouse clicks, etc. (These special methods are called 'hook methods'. [2])
- it provides an architecture that makes adding standard Macintosh UI features like undo, cut & paste, publish & subscribe, standard file dialogs, context-sensitive cursors, etc., very easy to do.

- it provides a widget set of standard Macintosh controls (for example, radio buttons, click boxes, and scroll bars).

With this definition of an application framework, both MacApp and the CPX ABCs are application frameworks, and so are THINK's TCL [3], Microsoft's MFC, and MetroWerks' PowerPlant. A more comprehensive overview of frameworks can be found in [4].

Since the whole point of an application framework is to make the task of building applications easier, the typical measure of a framework's success is the number of 'shipping' applications that have been built with it. (While this seems like such a common sense approach to measuring the success of a tool that it should almost go without saying, I have not always found this to be the case. At an academic conference on User Interface Management Systems – a research approach to generating GUIs automatically that has not yet seen commercial use – one smart aleck remarked that the differences between frameworks and UIMSs is that frameworks measure their success in apps/year whereas UIMSs measure their success in dissertations/decade. The really surprising part is that this remark was intended to poke fun at the lack of real-world UIMS success after many years of research, but the UIMS researchers thought that this measure was an excellent one, and accurately gauged their contributions. There is just no insulting some people.) *{OK, OK, I'll confess. The above-mentioned smart aleck was me.}*

Quick Comparison of MacApp and Prograph

	MacApp 3.0	Prograph CPX 1.1
Classes	206	340[1]
Methods	2127	2492[1]
Release	3.1	1.1
Implementation Language	C++	Prograph
Develop on:	Mac	Mac
Deliver on:	Mac PowerMac (native)	Mac PowerMac (native)[2] Windows[3]
"Shipping" Apps	Hundreds	Dozens

1 Includes Editor classes used only in Interpreter.
2 Expected in early Summer '95
3 Expected in late '95; Demonstrated at 1994 Prograph Conference

Table 1. A quick comparison of the gross characteristics of MacApp and Prograph CPX.

Are MacApp and CPX Comparable?

Table 1 shows the results of a comparison of the gross characteristics of MacApp and CPX. This quick comparison shows that these two frameworks are roughly comparable in size and targets, although MacApp, being more mature, has a considerably larger set of shipping apps.

Detailed Comparisons

Intro to MacApp

I assume that all the readers of MacTech have some knowledge of Prograph (the language, the development environment, and the ABCs) from earlier articles (for example, [5]). However, I don't assume any previous experience with MacApp, so let me give a short intro to MacApp, from the perspective of a Prograph CPX user. (If you want a more detailed intro to MacApp, see [2] or [6].)

MacApp is a C++ Macintosh application framework consisting of over 200 classes and 2100 methods. Since it is a framework supporting the generation of Macintosh applications, many of the classes in the framework reflect the basic Macintosh UI concepts: TApplication, TDocument, TWindow, TButton, TClipboardView, and so on, and the run-time interconnections between these objects reflect the structure of most Macintosh-style apps: the single TApplication object maintains a list of all the TDocument objects which in turn maintain lists of their TWindow objects which also maintain lists of their window widgets. In addition, there are classes that represent common abstractions like views (TView), user requests for action (TCommand), and frequently used building blocks (TList).

Any typical MacApp application will have a single subclass of TApplication, one or more subclasses of TDocument, zero or more subclasses of TWindow, and one or more subclasses of TView and TCommand respectively. These subclasses will override the hook methods – all of which have the common naming convention of starting with "Do", as in "TDocument.DoMakeWindow" and "TView.DoMouseCommand". In these regards, MacApp and the CPX ABCs are very similar.

MacApp has always been a Macintosh-only framework: compile on the Macintosh for the Macintosh. Last May (May 94), MacApp 3.1 shipped with support for the generation of PowerPC native applications. And MacApp 3.5, with support for OpenDoc™ container apps, is now in alpha release.

There are about 200 or so shrink-wrapped apps implemented with MacApp. Of these, Adobe PhotoShop is the best known, but others include Ray Dream's suite of high-end rendering applications, Intelligenetics GeneWorks, Spyglass Dicer, Apps MicroTV, Exis MacDSS, Odesta GeoQuery, MacTerminal, and the Apple developer tools ViewEdit and BalloonWriter. Like other frameworks, MacApp is often used for in-house applications, so its true success is often underrated.

MacApp ships with about ten example applications, all provided in source code. These range in complexity from the smallest possible MacApp application, Nothing, to a full spreadsheet application, and a small structured graphics editor. All MacApp users read, copy, and are influenced by the source code of these samples.

There are a variety of separate development tools associated with MacApp – browsers, window layout tools, source-level debuggers, run-time inspectors, etc. Some of these are provided by Apple (MPW, ViewEdit, SourceBug, and

APPLE WORLDWIDE DEVELOPERS CONFERENCE

FORCES OF CHANGE

In the beginning was "apple." As we were taught in school, it became the exemplar of the starting point... the foundation for our education. Then came Apple Computer, the new exemplar for looking at the world and the way we process information. Apple was the first to say, "Begin here and go forward into the future." They laid the foundation for change.

This year, the forces of change in computers and software will converge at the 1995 Apple Worldwide Developers Conference (WWDC). The industry's most exciting and far-reaching changes lie just ahead. Join the forces of change.

SAN JOSE CONVENTION CENTER

MAY 8-12, 1995

The WWDC: Where It All Begins

The Apple Worldwide Developers Conference is where foundations are laid and remarkable breakthroughs come to light. It is the major event of its kind. If something new is happening, chances are you'll see it in action or hear about it first at the WWDC.

Should You Attend?

Yes! Especially if you're a software engineer, consultant, systems integrator, multimedia developer, or technical manager. *Anyone breaking new ground in developing products or implementing customer solutions using Apple technologies should reserve a place at the WWDC without delay!*

What You'll See And Do

A lot. This year's conference again includes all-you-can-absorb technical sessions, plus a wider variety of breakout and hack sessions. You'll also find more about hot topics like RISC, OpenDoc, and Mac™OS.

What Else Is New

Learn about Apple's strategic direction in business and technology. Get the inside scoop on specific directions for the future and successful implementations of current technologies. Try out the latest interactive breakthroughs. Test your latest product ideas in compatibility labs staffed by Apple engineers.

Conference Information

The latest information on the events and activities surrounding the 1995 WWDC is available electronically through fax-on-demand information service, AppleLink and now the web. You can access complete conference information via:

- WWDC fax-on-demand information service:
  800-770-4863 in the U.S.
  International callers should call 415-637-2607 from the handset of their fax machine.

- AppleLink: HyperCard stack located in the WWDC folder. The pathname is:
  Developer Support; Developer Services; Events/Marcom; WWDC

- The dedicated conference web site:
  http://wwdc.carlson.com

MacBrowse, to name a few) and others, often competing tools, are offered by third parties (e.g., IcePick, Object Master, The Debugger, and CodeWarrior.) Most MacApp developers feel compelled to become proficient in many of these tools. Unfortunately, the languages and user interface conventions of the tools are not uniform.

Re-implementing the MacApp Samples

The project I decided to use to do a detailed comparison between MacApp and CPX was to re-implement some of the MacApp samples. This enabled me to test a wide variety of CPX features in a number of different contexts, yet was a project of reasonable size (especially important in an 'on-your-own-time' coding effort). While this project has proven to be very illuminating – both about MacApp and about CPX – it is not a flawless approach for an in-depth comparison. Flaws include:

- Size. The MacApp samples range in size from tiny toy apps to small apps. Even the largest (Calc) is only about 12,000 lines of C++. Thus, this re-implementation project really doesn't test the ability of CPX to build full size commercial products. (Of course, re-implementing PhotoShop in Prograph just to see if it can be done is a little more than I am willing to do just for fun.)
- Feature coverage. The MacApp samples were designed to demonstrate and test the features of MacApp, and thus they may not necessarily be representative of the types of apps that CPX is designed for. (At the very least, I should attempt the reverse test: re-implementing the CPX samples in MacApp. I have not done this since once I had done a significant amount of Prograph programming, it is *very* difficult to work up the motivation to return to the murky depths of C++, MPW, or even text-based programming at all. So, my work may be slightly one sided, but it is better than having no data at all.)

I have re-implemented four of the MacApp samples in CPX: Nothing, DemoText, IconEdit, and DemoDialogs. I used the built MacApp samples as runnable specifications for what I would implement in Prograph, and I purposely did not look at the MacApp code to see how some feature was implemented. (I have been a MacApp programmer for years and have studied the implementations of all the MacApp samples to some degree at some time or another. However, I didn't specifically re-examine the code while I was doing my Prograph re-implementation.)

The first two of these samples, Nothing and DemoText, were extremely easy and the re-implementation effort mainly amounted to re-arranging menu items on the standard Prograph menus to match those of the MacApp samples. Primarily, the re-implementation task for these two samples was easy because the functionality of these apps is so small, and they are direct uses of classes like TTEView (in MacApp) or EditText (in Prograph) so there is very little to do but 'wrap' these classes in an application. Secondarily, the re-implementation was easy since there were Prograph samples that were quite similar in functionality, and all I had to do was modify and extend them slightly to match the functionality of their MacApp counterparts.

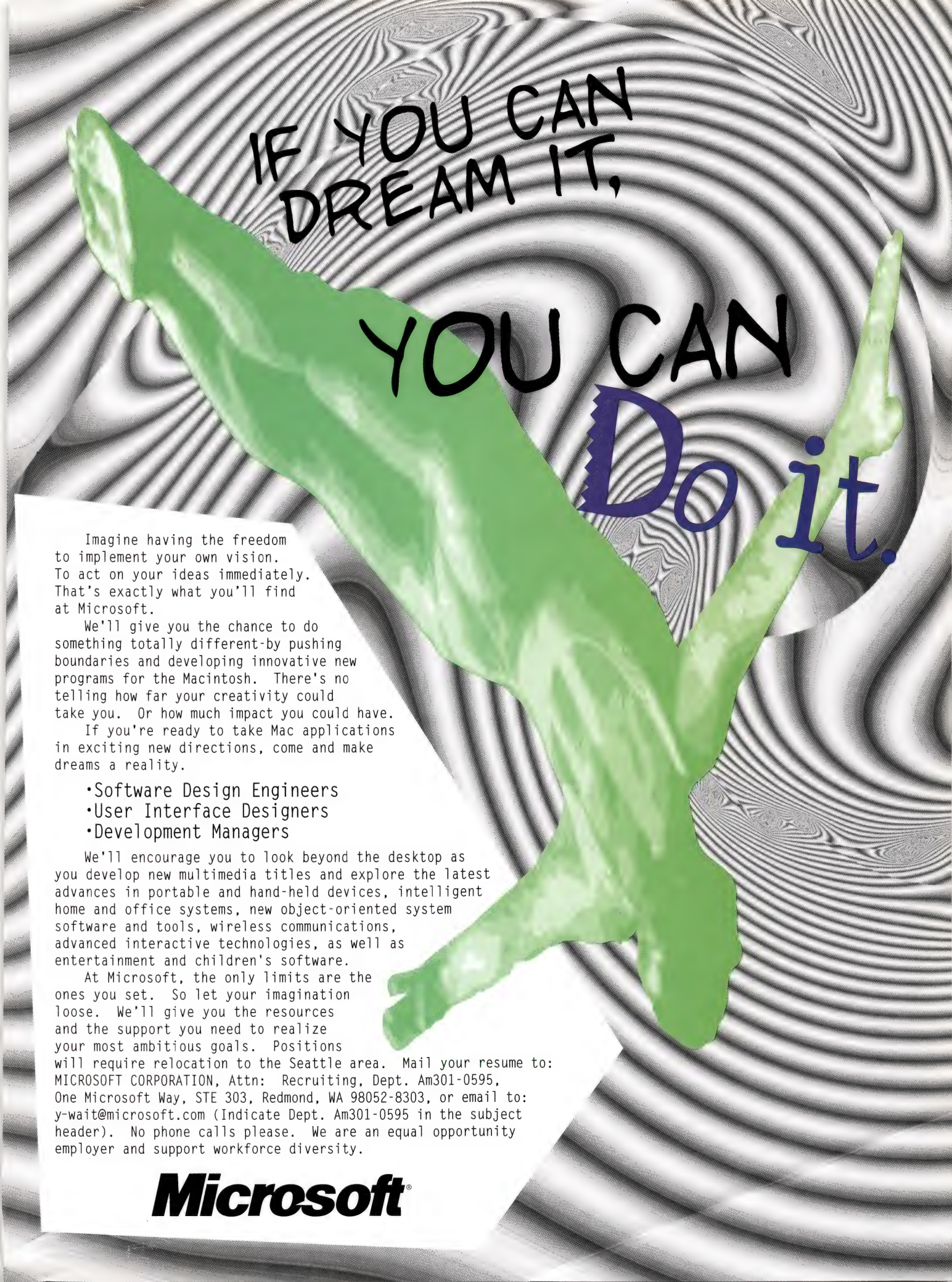
Re-implementing IconEdit was more code, but once I had fixed a small bug in Prograph's undo handling [7][8], it was not too difficult to get all the IconEdit functionality. One portion of the app that is tricky to implement in both versions is the handling of the editing of the icon, in particular, making sure that the user can mouse down in the fatbits view, move the cursor around while drawing, and then mouse up. Of course, choosing undo should then remove all the drawing added between the mouse down and the mouse up, not just undo the setting of the last drawn bit. Here is the C++ code from the MacApp version:

TIconDrawCommand::TrackMouse:

```
#pragma segment ADoCommand
pascal TTracker* TIconDrawCommand::TrackMouse(
          TrackPhase aTrackPhase, VPoint&   /*anchorPoint*/,
          VPoint& previousPoint,
          VPoint& nextPoint, Boolean mouseDidMove)
{
  VPoint fromBit, toBit, iconBit;
  short  lineLength;
  float  deltaH, deltaV;
  float  h, v;

  if (mouseDidMove) // If mouse moved since last time...
  {
    // Convert nextPoint and previousPoint to bit locations in the icon...
    fIconEditView->PointToBit(
      fIconEditView->ViewToQDPt(nextPoint), toBit);
    fIconEditView->PointToBit(
      fIconEditView->ViewToQDPt(previousPoint), fromBit);

    if (aTrackPhase == trackPress)
    {
      fOriginalIcon = fIconBitMap->Copy();
        // Make a copy of the original.
      fTurnBitsOn =!fIconBitMap->GetBit(toBit);
        // Turn bits on or off?
    }
// The following sets bits in the icon from the bit at previousPoint
// to the bit at nextPoint.  It is based on a simple line-drawing algorithm.
    lineLength = (short) Max(  labs(toBit.h - fromBit.h),
                               labs(toBit.v - fromBit.v));
    lineLength = (short) Max(1,lineLength);
      // This is on two lines so that Max can be "inlined"

    deltaH = (toBit.h - fromBit.h) / lineLength;
    deltaV = (toBit.v - fromBit.v) / lineLength;
    h = fromBit.h + 0.5;
    v = fromBit.v + 0.5;

    for (short i = 0; i < lineLength; i++)
    {
      iconBit.h = h;
      iconBit.v = v;
      fIconBitMap->SetBit(iconBit, fTurnBitsOn);
        // Set the bit in the icon
      fIconEditView->DrawBit(iconBit, fTurnBitsOn);
        // ...and draw it in the edit view.
      h = h + deltaH;
      v = v + deltaV;
    }
  }
  return this;  // Return same command object.
} // TIconDrawCommand::TrackMouse
```

TIconDrawCommand::TrackConstrain:

```
#pragma segment ADoCommand
pascal void TIconDrawCommand::TrackConstrain(
                    TrackPhase      /*aTrackPhase*/,
                    const VPoint&   /*anchorPoint*/,
                    const VPoint&   /*previousPoint*/,
                    VPoint&           nextPoint,
                    Boolean         /*mouseDidMove*/)
{
```



Imagine having the freedom to implement your own vision. To act on your ideas immediately. That's exactly what you'll find at Microsoft.

We'll give you the chance to do something totally different-by pushing boundaries and developing innovative new programs for the Macintosh. There's no telling how far your creativity could take you. Or how much impact you could have.

If you're ready to take Mac applications in exciting new directions, come and make dreams a reality.

- Software Design Engineers
- User Interface Designers
- Development Managers

We'll encourage you to look beyond the desktop as you develop new multimedia titles and explore the latest advances in portable and hand-held devices, intelligent home and office systems, new object-oriented system software and tools, wireless communications, advanced interactive technologies, as well as entertainment and children's software.

At Microsoft, the only limits are the ones you set. So let your imagination loose. We'll give you the resources and the support you need to realize your most ambitious goals. Positions will require relocation to the Seattle area. Mail your resume to: MICROSOFT CORPORATION, Attn: Recruiting, Dept. Am301-0595, One Microsoft Way, STE 303, Redmond, WA 98052-8303, or email to: y-wait@microsoft.com (Indicate Dept. Am301-0595 in the subject header). No phone calls please. We are an equal opportunity employer and support workforce diversity.

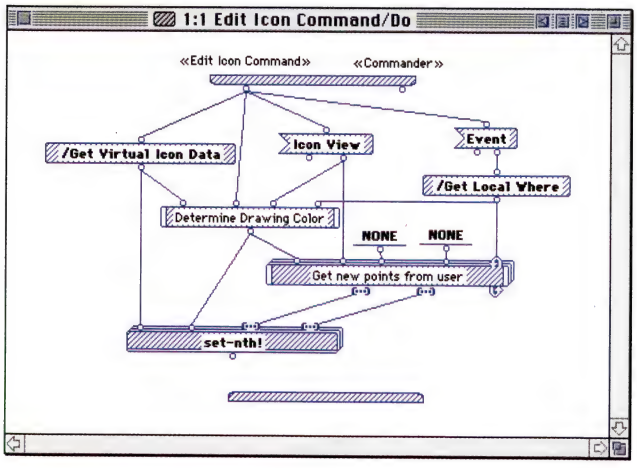
Microsoft®

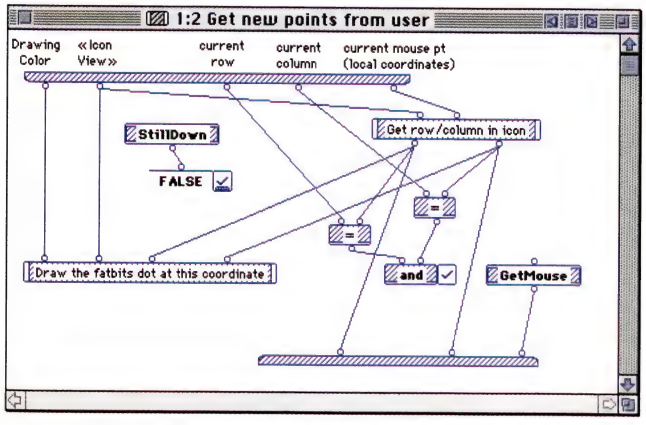
```
    // This is on several lines so that Max can be "inlined"
    VCoordinate h = Min(nextPoint.h,
            fIconEditView->fSize.h - kBorder - 1);
    VCoordinate v = Min(nextPoint.v,
            fIconEditView->fSize.v - kBorder - 1);
    h = Max(h, (long) kBorder);
    v = Max(v, (long) kBorder);
    nextPoint = VPoint(h, v);
} // TIconDrawCommand::TrackConstrain
```

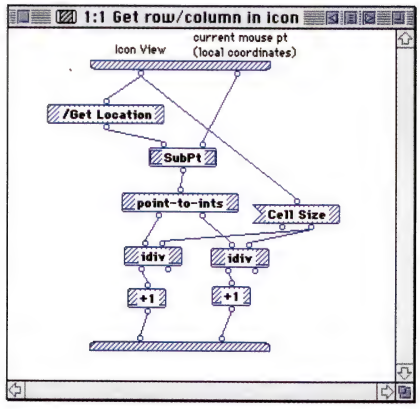
TIconDrawCommand::DoIt:

```
#pragma segment ADoCommand

pascal void TIconDrawCommand::DoIt()
{
    VRect editViewExtent;

    fIconDocument->SetIcon(fIconBitMap); // Set the document's bit map.

    // SetIcon invalidates all views of the document, including the one we just drew in.
    // To avoid flashing, validate the edit view here so it doesn't get redrawn.

    fIconEditView->GetExtent(editViewExtent);
    fIconEditView->ValidateVRect(editViewExtent);
} // TIconDrawCommand::DoIt
```

and Figure 1 shows the approximately equivalent Prograph code (together with some of its scaffolding, so that you can more easily see what is going on.) Perhaps it is my visual coding bias, but I find the Prograph code much easier to read and quicker to understand.







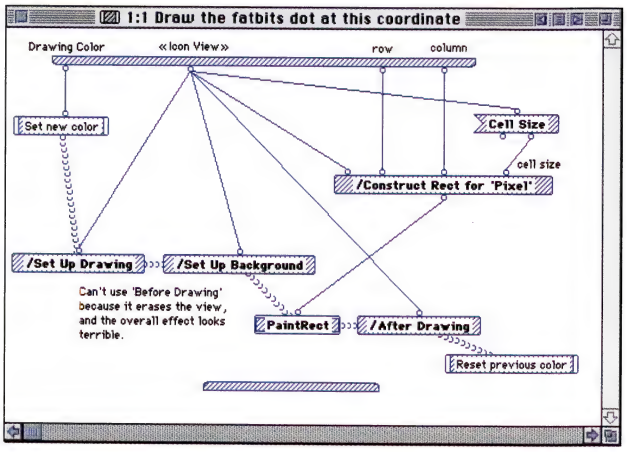


Figure 1. The implementation of fatbits drawing for the IconEdit sample program re-implementation in Prograph.

I also took the opportunity while doing the IconEdit re-implementation to add Filters & Sieves to the Prograph application framework – something I had always wanted to do with MacApp, but was never able to work up the courage to delve that deeply into the inner working of MacApp. [9] With Prograph, and especially because of its excellent editor/interpreter/debugger environment, I was able to accomplish this, and enjoyed doing it.

DemoDialogs was a much bigger effort. The MacApp version is about 3500 lines of C++ and is basically a collection of all the window types and widgets that MacApp supports, and demonstrations of how to build your own if MacApp doesn't provide them directly. DemoDialogs is the kind of sample program that is actually a collection of small samples: you look up a portion of the source code to see how *that* single feature was implemented. (Other types of sample programs include those that are good starting points for certain types of applications (e.g., DrawShapes or Skeleton), those that are just tours de force of the power of the framework (Nothing), or those that are tutorial in nature (IconEdit)). The MacApp DemoDialogs has example windows of the type that are seen in every Macintosh application (save dialog, print dialog, etc.), commonly used windows (e.g., a font picker like you would see

ScriptX Language Kit Version 1.0 Now Available!

ScriptX™

Ships!

Version 1.0 Includes:

- Free Upgrade to Version 1.1
- 60 Day Warranty With A "No Questions Asked" Return Policy.

Are you ready to expand your creative range? If so, then join the Founding Kaleida Vanguard & Associate developers and experienced programmers, who are ready to revolutionize this industry by delivering the next generation of multimedia products using the ScriptX Language Kit™.

Contact Kaleida Direct:

800·6KALEIDA • 415·335·2098

Internet: kaleida.direct@kaleida.com, or http://www.kaleida.com/

Be part of the NEW Kaleida Worldwide Developer Program and gain access to:

- **Developer Technical Support**
- **ScriptX Training Courses**
- **Customized Consulting Services**
- **Informational Support and Co-marketing Opportunities**

KALEIDA LABS SCRIPTX
Kaleida Worldwide Developer Program™

Kaleida Labs

in any word processing application), some windows that you see occasionally (windows with custom controls, windows with arrays of buttons and other widgets), and then some windows that you never see, but show off some feature of MacApp (a View inspector that shows you the internal structure of any other window in DemoDialogs). The feeling that you get after using DemoDialogs for a little while is that MacApp will probably be able to build any type of window that you want, and it may even provide direct support for that kind of window.

The final result of my DemoDialogs re-implementation effort is that I was able to implement all the functionality of DemoDialogs except for only one small, but possibly significant feature: large views. (Discussed in detail below.) Figure 2 shows all the windows of the CPX DemoDialogs. (A similar figure for the MacApp implementation would be exactly the same, so I have not included it here.) The Prograph implementation of DemoDialogs, like the MacApp original, gives you the feeling that you can create any type of window with CPX. The CPX re-implementation of DemoDialogs is perhaps a little bit slower, but comparable in perceived responsiveness to the MacApp version. The Prograph version does have a somewhat little larger RAM footprint. (See the detailed measurements on RAM and disk footprints later in this article.)

The single feature of DemoDialogs that is not easily implementable in Prograph is a 10,000 element scrolling list. Like MacApp 1.0, Prograph uses standard 16-bit QuickDraw coordinates for its views. This limits the size of views. MacApp 2.0 (and later) went to 32-bit coordinate representations for all view objects, including scrolling lists. The result was that the biggest scrolling list that I could build in Prograph was about 1,000 elements in length. Whether this is significant to you or not, depends in the types of applications you want to do. Details of the DemoDialogs re-implementation are discussed in [10].

Which is Better, MacApp or CPX?

There is no quick easy answer to the question of which of these two frameworks is better since both products are significant, feature-rich frameworks that provide great utility to their users. In addition, there are many of aspects of both MacApp and CPX that this sample re-implementation test doesn't stress. For example, both MacApp and CPX support far code and far data – features that will only really be used in larger applications (greater than 4000 methods or more than 32K of global data). However, it is possible to list some of the aspects of each that are better than those of the other framework.

What's Better in MacApp?

Sample Programs. The MacApp sample programs are significantly broader and deeper than those in Prograph, and there are more of them.

Clipboard View. While both MacApp and CPX support the Macintosh clipboard for copying and pasting of text and pictures, MacApp also implements a clipboard view so that you can trivially add a "Show Clipboard" menu item to your application. I also found the MacApp approach to providing clipboard support for your application's data types (TApplication.MakeAlienViewforClipboard) to be much easier to understand and to use. (I do wish, however, that MacApp's clipboard supported more than just text and pictures, but also included QuickTime movies, sound, etc.)

Low memory handling. MacApp's mechanism for dealing with low memory situations is more robust and more useful than Prograph's Rainy Day Fund.

Publish and Subscribe. MacApp provides a framework for supporting the Macintosh publish and subscribe mechanism (actually, it is a unified framework for the clipboard as well as P&S). MacApp handles a lot of the details of the P&S user interface (borders, standard dialog boxes, etc.) for you.

Adorners. MacApp's mechanism for adding some drawing adornments to views is better at this task than the borders mechanism is in CPX. For one thing, the adorners mechanism enables you to add multiple adorners to a single view.

What's Better in CPX?

Language. Prograph is a more readable, simpler language than C++, yet it is just as powerful. In fact, it isn't possible to write syntactically incorrect Prograph code, and semantic errors often jump out at you in even the most casual examination. The clarity of code expressed in Prograph is due, in part, to the fact that algorithms, by their very nature, have an inherent internal structure. Often, implementing an algorithm in a textual, and thus necessarily linear fashion, hides this structure. In contrast, the Prograph language brings this structure to the surface and enables the Prograph-literate programmer to grasp this structure in a single gestalt. Prograph programmers also don't need to spend time thinking up descriptive names for temporary variables in their code, they just draw a data flow line between the point where the data is generated and where it is used. Data is typed dynamically and this makes both prototyping and significant architectural changes easier.

Environment. Prograph has a fast garbage collector, an integrated source-level debugger – which lets you change code or data while your application is running – and an extensible window and view editor.

Behaviors. Behaviors provide a much easier to use, and much more flexible way for your code to be accessed by the framework than MacApp's hook methods. With a behavior, you are no longer constrained to the parameter list of the hook method. In addition, CPX will retrieve some data for you (like getting values out of text items in dialog boxes.) Consider, for example, the task of responding to a click on a button in a dialog box. Here is what you have to write in MacApp:

TModelessBeepDialog::DoEvent:

```
#pragma segment ARes
pascal void TModelessBeepDialog::DoEvent(
                    EventNumber     eventNumber,
                    TEventHandler*  source,
                    TEvent*         event)// override
{
  long i;
```

What's New With ***AppMaker***?

Even more customizable and extensible—The Source Code Generator, the Resource Generator, and the User Interface Editor are soft-coded and user-definable.

Designed for portability—A single AppMaker document can now generate code and resources for a wide variety of languages, frameworks, and eventually, platforms.

Supports CodeWarrior PowerPlant, NeoAccess, and PowerMac development.

New features include an outline view, a simulator, and support for color.

Includes one-year automatic update service with at least three CDs per year.

(By the way, the b2 release is 5-times faster than the b1 release.)

B • O • W • E • R • S
Development

97 Lowell Road, Concord, MA 01742
(508) 369-8175 • FAX 369-8224

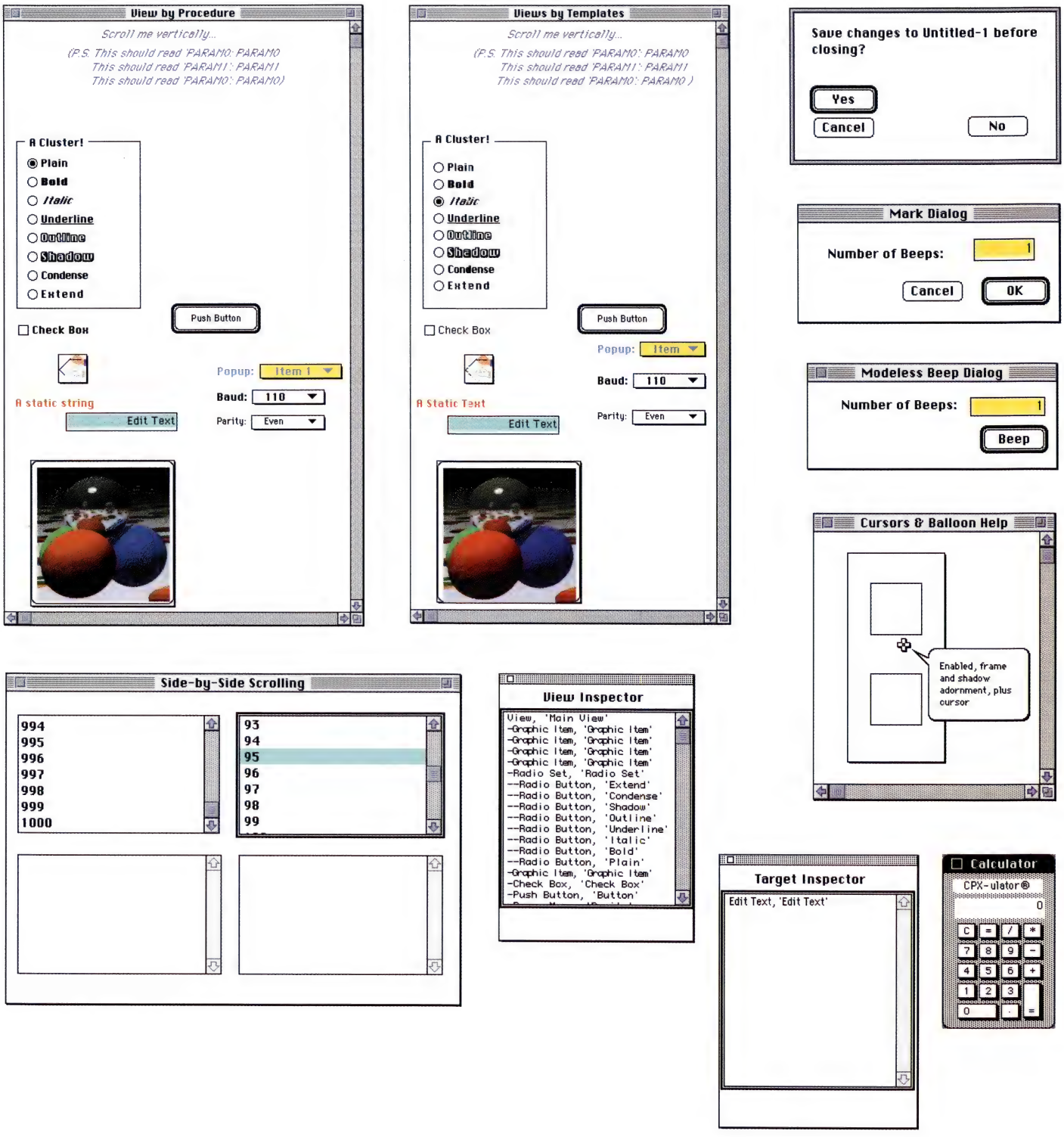
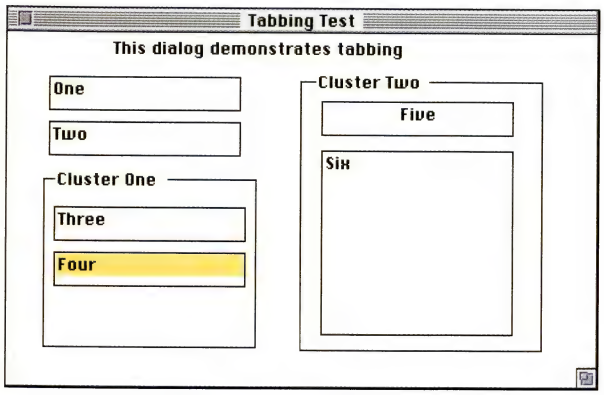
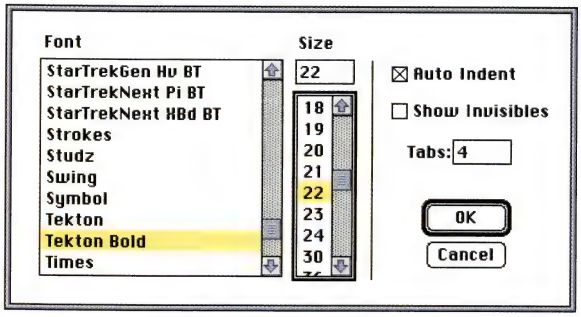
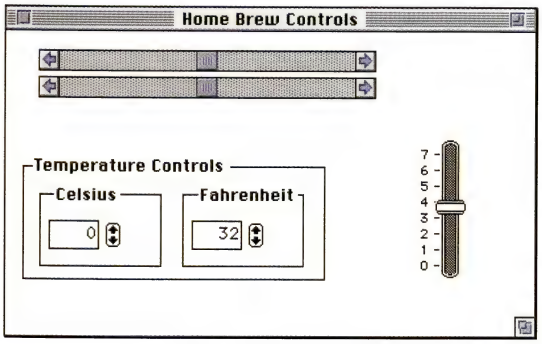
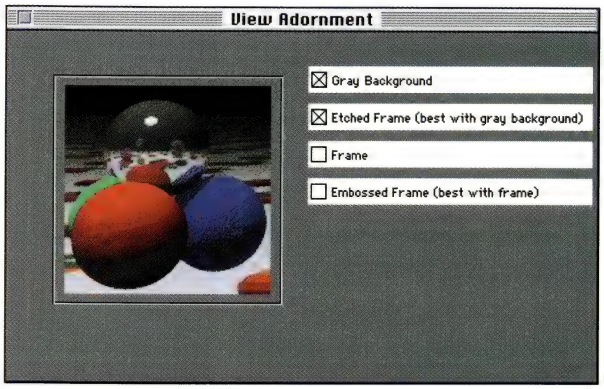


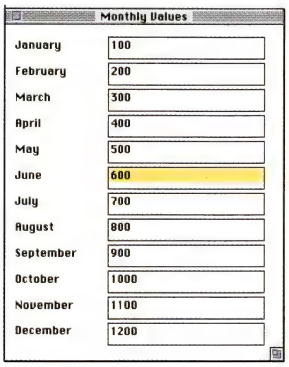
Figure 2. The windows of the CPX implementation of DemoDialogs. Note that a similar figure of the windows of the MacApp implementation would be virtually identical.

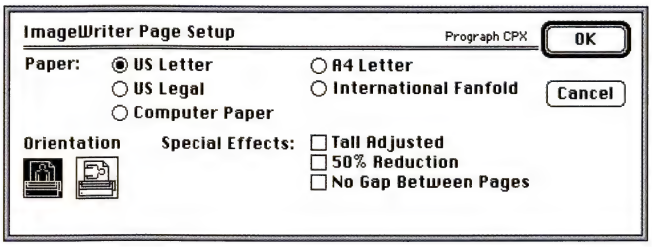


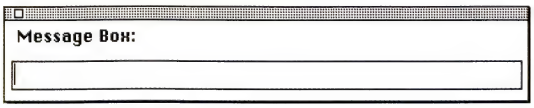




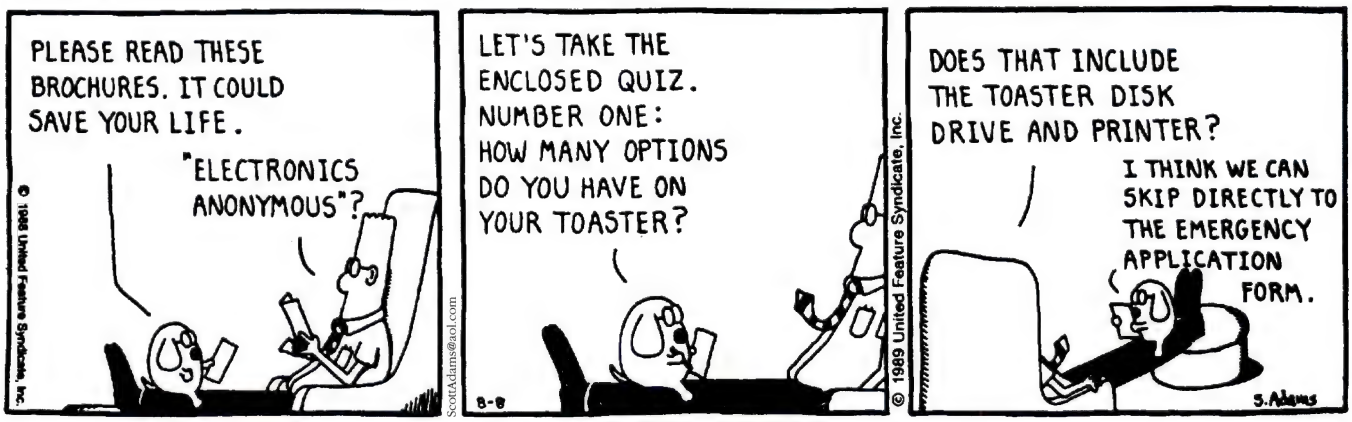








Dilbert® *by Scott Adams*



reprinted by permission of UFS, Inc.

```
    long maxi;
    switch (eventNumber)
    {
      case mButtonHit:   // Default button and Enter or Return key
        {
          maxi = ((TNumberText *)
                  (FindSubView('numb')))->GetValue();
          if (this->IsHierarchyValid())
            for (i = 1; i <= maxi; i++)
              gApplication->Beep(2);
          break;
        }
      default:
        inherited::DoEvent(eventNumber, source, event);
    }
} // TModelessBeepDialog::DoEvent
```

Figure 3 shows the simple dialog you have to fill out, and the small amount of code that you have to write to achieve the same result in CPX.

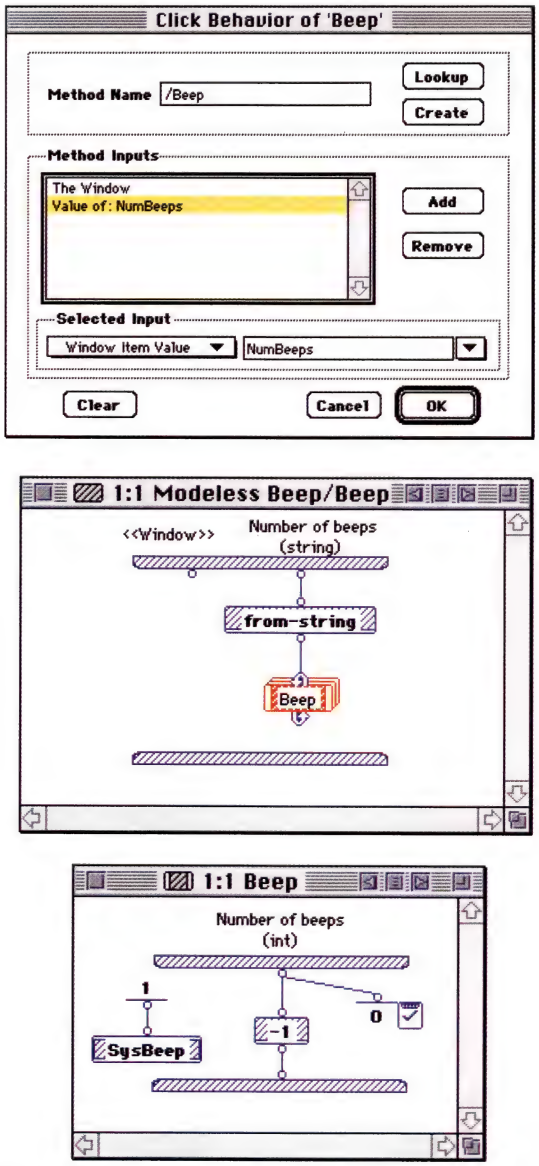


Figure 3. Handling a mouse down in a button is easy because of Prograph Behaviors.

Menus. For the MacApp programmer, menus exist at the boundary between the object world of MacApp and the lower-level world of the Toolbox. Some of the programming tasks are in one world and language (DoSetupMenus) and some in another ("Arrow", noIcon, noKey, noMark, plain, cArrow). Having taught many MacApp classes, I can testify that this can be a considerable stumbling block to the new MacApp user. Thus, in the total task of adding a new menu or menu item to a MacApp application, you have to constantly flit back and forth between a number of different programming tools each with its associated language(s) and style of usage. You code in an object language to implement the main functionality of your app, in a strange C-like language (Rez) to describe your menus, and in even stranger language (MPW Shell) to build your application. Along the way you have to use separate applications (e.g., ViewEdit and ResEdit) to build in a direct manipulation style some of the objects that your menus will operate with or on.

CPX changes this dramatically by making menus and the menubar into objects that you deal with in exactly the same way that you deal with other objects like windows and views. The Menus section in the CPX ABCs provides you with eight classes associated with menus, with the classes Menu, Menubar, and Menu Behavior among them. These classes provide you with methods that add or remove menus from the menubar, add or remove items from an individual menu, enable a menu item, test to see if a menu item is checked or enabled, and find an individual menu or menu item, among other things. You invoke direct manipulation object editors for the menubar and menu objects in exactly the same way that as for windows or views or any other object. There isn't any need to use another separate tool to accomplish a critical portion of the basic task of implementing the menu functionality of your app. Internally, CPX still deals with the Toolbox Menu Manager and uses the Macintosh ROM to implement menus. However, all the Toolbox details are abstracted for you by CPX and you deal with menus as objects.

This small change of moving menus from the Toolbox to the framework dramatically simplifies both the programmer's model and the task of getting that first application up and running. Just as important, abstracting away the Toolbox details simplifies porting your CPX source code to other platforms.

Offscreens. Getting smooth screen refreshes in MacApp is tricky and involves buying a third party product (OSImage) or rolling your own code in support of GWorld and offscreens. In CPX, all you have to do is allocate an Offscreen object (part of the CPX class library) and attach it to a window. Nothing could be simpler.

Simple tasks are simpler. I have found on many occasions that a number of conceptually easy tasks (opening a window, responding to a menu item choice, etc.) are just much easier in CPX than in MacApp. Here is a case in point that brings together many of these: posing a dialog. First the MacApp way:

SONY

Some Ideas Come And Go.

Trends are fleeting. But true innovation endures. Sony pioneered mass storage technologies and peripherals. Rewritable MO, DDS tape, CD-ROM drives, our new recordable CD-ROM, floppy drives, and the revolutionary MD Data. High performance. Ahead of their time. To learn more about Sony Data Storage Products, call 1-800-352-7669 (ext. 109) for the reseller nearest you. For ideas that transcend time, look to Sony.

Others Just Keep On Going.

TTestApplication::MakeModalBeepDialog:

```
pascal void TTestApplication::MakeModalBeepDialog(
                              CommandNumber aCommandNumber)
{
  TWindow *aWindow;
  IDType   dismisser;
  long     n;

  FailNIL(aWindow = gViewServer->NewTemplateWindow(
                        (short)aCommandNumber, NULL));
  dismisser = aWindow->PoseModally();
  if (dismisser == 'ok  ')
    n = ((TNumberText *)
              (aWindow->FindSubView('numb')))->GetValue();
  else
    n = 0;
  aWindow->Close();
  for (long i = 1; i <= n; i++)
    gApplication->Beep(2);
} // TTestApplication::MakeModalBeepDialog
```

Notice how many concepts are buried in the main line:

```
FailNIL(aWindow = gViewServer->NewTemplateWindow(
                      (short)aCommandNumber, NULL));
```

The view server, window templates, the MacApp failure handling mechanism must be understood, and you have to tell the C++ compiler *again* that the first parameter to a standard MacApp utility routine is an integer. Contrast this with the CPX way shown in figure 4. *Much* simpler. However, even this code is not the simplest possible. Dan Shafer suggested to me an even simpler way shown in Figure 5 in which you write no code at all. What could be easier?

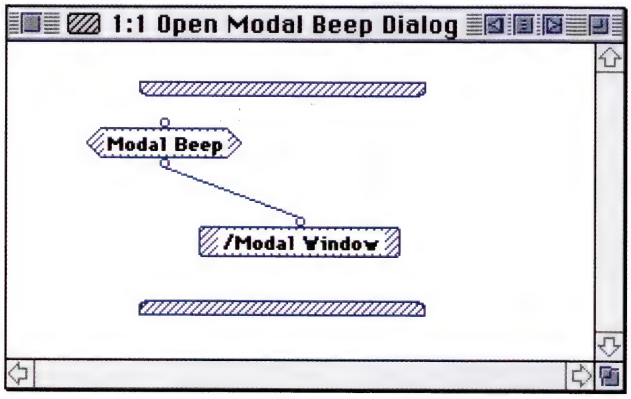


Figure 4. Displaying a modal dialog in the Prograph CPX framework. The equivalent code in MacApp is:

```
FailNIL(aWindow = gViewServer->NewTemplateWindow(
                                (short)aCommandNumber, NULL));
dismisser = aWindow->PoseModally();
```

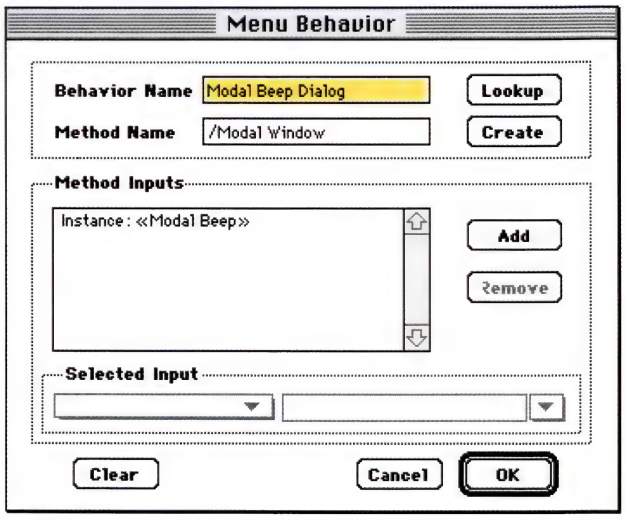


Figure 5. While the Prograph code in figure 4 looks simple, here's an even simpler way.

Measurements

I have been able to gather some objective data about RAM and disk footprints, and performance of one easily timed interaction.

RAM and Disk Footprints

Figures 6 and 7 show the RAM and disk footprint comparisons of four of the MacApp samples and my re-implementations of them. Note that there is a constant increment for the Prograph versions of about 200-300K in the case of disk footprints and 300-400K in the case of RAM footprints. A little more than half of this increase can be accounted for by the additional symbol tables that Prograph must maintain so that it can do by name dispatching at run-time ('inject' in Prograph terminology).

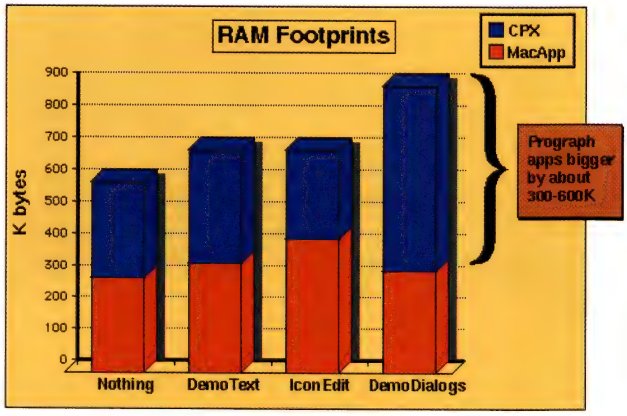


Figure 6. A comparison of the RAM footprints for four of the MacApp samples, and their re-implementations in Prograph.

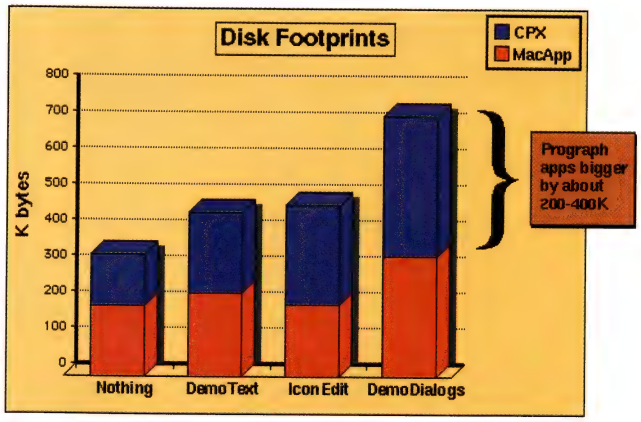


Figure 7. A comparison of the disk footprints for four of the MacApp samples, and their re-implementations in Prograph.

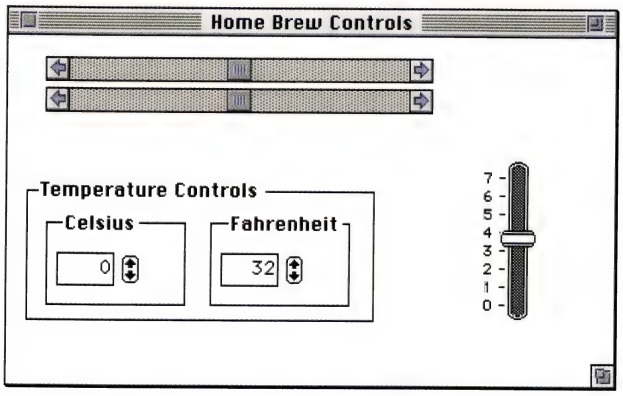


Figure 8. The custom temperature conversion control, from the DemoDialogs sample, that was used for a speed test between MacApp and Prograph.

What can't be determined from this data is whether these increments will be remain constant as the size of the application grows, or whether they too will grow. If they are constant, then an additional 400K memory hit is not too significant when the app is expected to require a 6MB RAM partition. An examination of the RAM footprint of shipping Prograph applications shows that they do not have exorbitant RAM requirements, which suggests that the Prograph RAM increase is a constant, or a very slowly growing function.

Performance

Measuring interaction performance objectively can be difficult. Subjectively, as I already noted above, MacApp apps seem slightly snappier but approximately comparable with their Prograph counterparts.

I was able to find one interaction where objective timing data could be gathered. In the DemoDialogs application, there is a Celsius-Fahrenheit control in which the user can dial in a temperature on one scale and see the value in the other scale displayed. (See figure 8.) The dialing controls respond to mouse clicks with single increments or decrements, or if the mouse button is held down, the numbers increment as quickly as possible with all intermediate numbers displayed. This suggested to me the following test: Hold the mouse button down for five seconds in the increment Celsius control and see how large the Celsius edit text value gets in that fixed amount of time. This test measures ToolBox access (StillDown), conditional evaluation, integer arithmetic, integer to string conversion, and view refreshing. Table 2 shows the surprising results of this test. Does this mean that Prograph is ten times the speed of C++? No, it doesn't. What this means is that language speed differences are dominated by better algorithmic choices, especially for time-consuming operations like screen refreshing, and in this particular case Prograph has the better algorithm.

Interaction Speed Test
Home Brew Temperature Calculator from the DemoDialogs Sample
(Larger numbers are better)

	MacApp 3.0	CPX 1.1
Interpreted		222
Compiled	63	578

Each number is an average of three tests. All tests run on a Quadra 840AV running System 7.5.

Table 2. Timing test results between MacApp and Prograph.

In this test, the mouse button is pressed for 5 seconds over the Celsius increment control, and the Celsius edit text field increase as quickly as it can, displaying all intermediate values. The table record how many degrees were incremented in the 5 seconds. The surprising result is that interpreted Prograph is about four times as fast as compiled MacApp, and compiled Prograph is about nine times as fast as compiled MacApp. Note that this does not mean that overall a Prograph app will be nine times as fast as a MacApp one, but that some computations can be.

Overall

What more can I say than that Prograph CPX is has been my development environment of choice for producing applications for the past two years or so, and will probably remain so for the foreseeable future.

References

[1] Ken Doyle, Barry Haines, Mark Lentczner, and Larry Rosenstein, "An Object Oriented Approach to Macintosh Application Development", *Journées Langages Orientés Objet*, BIGRE +

GLOBULE, Nov. 1984, pp. 46-54.

[2] Schmucker, Kurt. *Object Oriented Programming for the Macintosh*, Hayden Book Company, 1986.

[3] Parker, Richard O. *Easy Object Programming for the Macintosh using AppMaker and THINK Pascal*, Prentice-Hall, 1993

[4] Lewis, Ted (ed.). *Application Frameworks*, Prentice-Hall, 1995.

[5] Schmucker, Kurt. "Prograph CPX", in *Application Frameworks*, Ted Lewis (ed.), Prentice-Hall, 1995; reprinted in MacTech Magazine, November, 1994, pp.69-80.

[6] Wilson, David A., Rosenstein, Larry S., and Shafer, Dan, *Programming with MacApp*, Addison-Wesley, 1990.

[7] Schmucker, Kurt. "A Tutorial on 'Undo' in CPX", *Visual News*, July 1994, pp. 7-12.

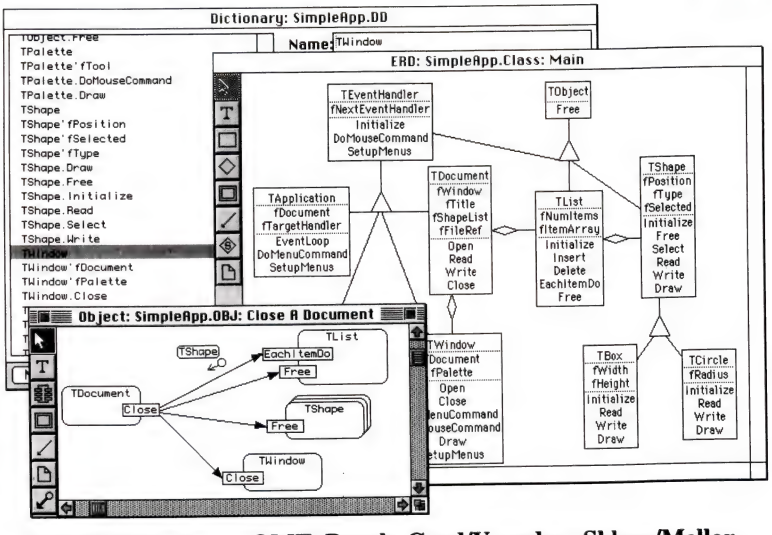
[8] Schmucker, Kurt. "Commands and Undo in Prograph CPX", *MacTech Magazine*, January 1995, pp. 18-31.

[9] Schmucker, Kurt. "Filters and Sieves in Prograph CPX", *MacTech Magazine*, March 1995.

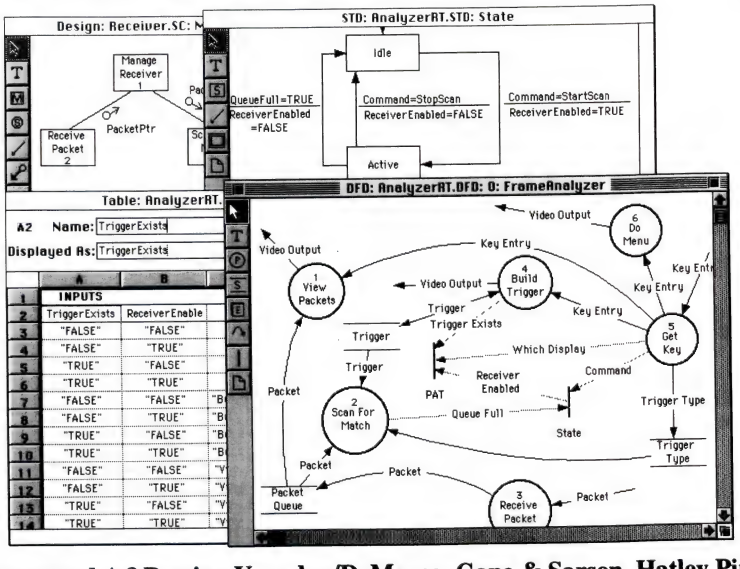
[10] Schmucker, Kurt. "DemoDialogs in Prograph CPX", *FrameWorks*, Volume 8, Number 2, (March/April 1994), pp. 8-13.

MacAnalyst and MacDesigner

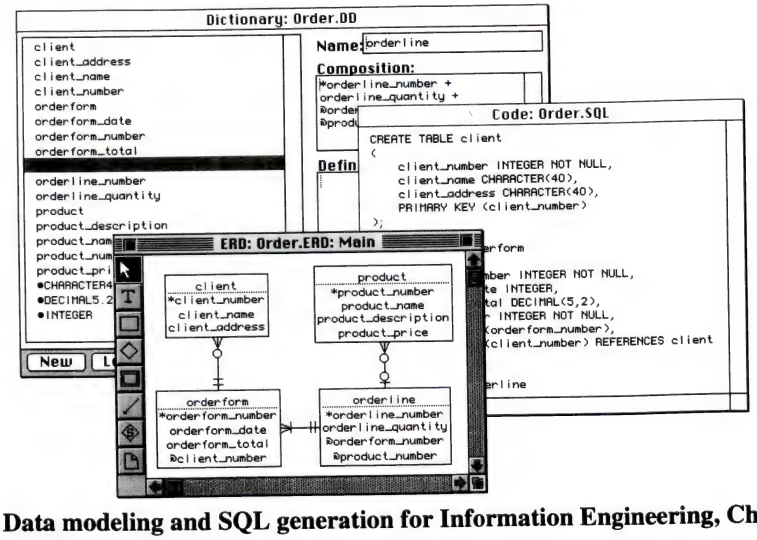
Computer Aided Software Engineering For The Macintosh



OOA/OOD includes OMT, Booch, Coad/Yourdon, Shlaer/Mellor...



Structured A&D using Yourdon/DeMarco, Gane & Sarson, Hatley Pirbhai...



Data modeling and SQL generation for Information Engineering, Chen...

Draw your design, generate code frames, or reengineer existing code back to design diagrams using an integrated tool that supports today's most popular methods.

Software Engineering

Structured Analysis & Design
Object-Oriented Analysis & Design
Real-Time & Task Design
Data & Information Modeling
Screen Prototyping
Integrated Code Editing & Browsing
Multi-User Dictionary & Requirements
Code to Design for C++, Pascal, Fortran...
Design to Code for C++, Pascal, Fortran...

Product Options

MacAnalyst	$ 995
MacAnalyst/Expert	$1595
MacDesigner	$ 995
MacDesigner/Expert	$1595
MacA&D	$2995
Translator	$ 495

Each product is available by single, site, or educational license and supported with on-site training, update service, free newsletter, and technical support.

System Requirements

Macintosh System 7 or A/UX 3 or later and 6 Meg RAM. Also runs on Unix machines with Apple's MAE software.

Winner of 1994 CIO Magazine Readers' Choice Award. Find out why thousands of developers use MacAnalyst and MacDesigner tools for personal computer, mainframe, and embedded software projects. Call today for free technical brochures!

Excel Software

515-752-5359

P.O. Box 1414 • Marshalltown, IA 50158
CASETOOLS@AOL.COM • Fax: 515-752-2435

Object EXPO

THE NATIONAL CONFERENCE & EXPOSITION

June 5-9, 1995

NY Hilton New York City

Setting New Strategies— Reaching New Goals

Object Expo returns to New York in 1995. It brings together the most respected experts and leading companies in the object technology industry. Whether you're just exploring possibilities, or are a seasoned professional, don't miss this once a year conference. It's the best place to learn the latest techniques, develop new strategies, and stay up-to-date on trends and breakthroughs in object technology.

Object Expo '95 Provides the Strongest Technical Program Available

This year's program offers extensive technical training through real-life case studies and experiences presented by a prestigious faculty of experts who motivate and move this industry.

Educational tracks include:

- C++
- Fundamentals
- Business Stategies
- Databases
- Analysis and Design
- Smalltalk
- User Experiences

NEW tracks focusing on:

- **Client/Server Development**
  Provides attendees with guidelines for business projects and a complete framework for implementing and managing client/server solutions.
- **Management Strategies Symposium**
  This separate 1/2 day event is geared for upper-level software managers exploring the benefits of OT adoption and implementation. Learn first-hand from your peers how others have successfully implemented object-technology to solve business problems.

Special Educational Events Include:

▲ Keynote Speeches
▲ Walk-In Clinics
▲ Product Education Sessions

Gather with your peers at unique seminars fueled by group participation:

▲ User Group Meetings
▲ Birds-of-a-Feather Sessions
▲ Panel Discussions

Be a part of the most high-powered OT event on the East Coast! Don't miss this conference and exhibition dedicated to setting new strategies with object technology that guide you to higher levels of software productivity.

Sponsored by:

OBJECT magazine
JOURNAL OF OBJECT-ORIENTED programming
C++ REPORT
THE Smalltalk REPORT

Presented by:

SIGS CONFERENCES

A Partial Listing of 1995 Exhibitors:

Application Development Trends ▼ Atelier Research ▼ Cadre Technologies ▼ CenterLine ▼ DEC ▼ DevelopMentor ▼ DHR Technologies ▼ Easel ▼ Expersoft ▼ Franz, Inc. ▼ Hatteras Software ▼ Hitachi ▼ IBM ▼ ICON Computing ▼ IDE ▼ INCONIX Software Engineering ▼ Liant Software ▼ Mactech Magazine ▼ Martin Marietta ▼ McCabe & Associates ▼ Miller Freeman ▼ Morgan, Parker and Johnson ▼ Netlinks Technology ▼ NetSmiths Limited ▼ Object Design ▼ Object International ▼ Objectory ▼ ObjectSpace ▼ Open Objects ▼ ParcPlace Systems ▼ Persistence ▼ Prentice Hall ▼ Proforma ▼ Project Technology ▼ Rational Software ▼ Rogue Wave Software ▼ Salient ▼ Select Software Tools ▼ Semaphore ▼ Servio ▼ SES ▼ SIGS Publications ▼ Synrix ▼ TakeFive Software ▼ Tower Technology ▼ Trinzic ▼ TRW ▼ UniSQL ▼ Waters Information Services

Please send me more information on

22ADMM **Object Expo**

❑ Attending Technical Conference ❑ Mgmt. Strategies Symposium ❑ Exhibiting ❑ Receiving Free Exhibits Pass

Name _____

Company _____

Address _____

City _____

State _____ Zip _____

Day Phone _____

Fax _____

Mail or Fax coupon to: Object Expo
71 West 23rd Street, New York, NY 10010
Fax: 212/242-7578

MAGIC

by Richard Clark, Scott Knaster, and the staff of General Magic

General Magic Magic Cap™

Making Magic

A developer's introduction to General Magic and Magic Cap

"Where should Apple go after the Macintosh?" The year was 1990, and Marc Porat was leading a group of engineers considering that question. Porat's answer to this question became the Paradigm project: Apple's plan to build a small, personal, communication-oriented computer. Porat asked to spin off a small company to realize this vision, Apple's board gave its blessing, and *General Magic* was born.

Several members of the original Macintosh team – Andy Hertzfeld, Bill Atkinson, Joanna Hoffman, and Susan Kare – joined with scores of new *Magicians* to realize Porat's dream. Major companies (including Apple, Sony, Motorola, AT&T, Matsushita, and Philips) formed an alliance around General Magic to bring their talents and resources to the project. Last October, the world saw the first examples of that work when Sony shipped the Magic Link™ Personal Intelligent Communicator and AT&T opened the AT&T PersonaLink™ Services network. Two months later, Motorola shipped Envoy™, which is their implementation of a Personal Intelligent Communicator. These products have two things in common: Magic Cap™ and Telescript™.

Even though this article is about Magic Cap, we should take a moment to describe Telescript. Telescript is a programming language for building *distributed systems*—programs which aren't entirely resident within a single computer but are spread out across a network. AT&T's PersonaLink Services use Telescript programs to carry messages between a communicator and the network, then uses these programs to transport messages around the network. Even though Magic Cap uses Telescript for many of its communications tasks, we'll see that there are plenty of opportunities that Magic Cap developers can pursue without Telescript.

"Magic Cap embodies the original spirit of the Macintosh – it creates an inviting world that uses familiar objects in common ways."

What is Magic Cap?

Magic Cap is General Magic's software environment for *personal communicators*. Magic Cap embraces both the most common tasks found in today's workplace (managing your schedule and relationships) and such communication tools as the telephone, FAX, and electronic mail. Magic Cap doesn't skimp on the programming side either: it contains an extensive collection of user interface building blocks, a multitasking kernel, and a rich set of utility routines for third-party developers.

Richard Clark and Scott Knaster – Richard is known to many as the guy who taught them a lot about the Macintosh, either at Developer University, at MacHack, or here in the pages of MacTech Magazine. He now does much the same job at General Magic, teaching developers all about this new platform. Scott Knaster is perhaps best known for his complete collection of Mad Magazines. You might also have seen some books he wrote, or heard of how he was at Apple in the days when the Macintosh was first launched, and was instrumental in creating, among other things, Developer Technical Support.

Magic Cap was designed with two key principles in mind: communication and ease of use. *Communication* means that every device is ready to communicate: all Magic Cap communicators have a data/FAX modem built-in along with the software to support it. Each communicator is also designed to act as a telephone dialer and can act as a handset with the appropriate attachment. Communicators may interact by sending electronic mail over the telephone or directly via an infrared transceiver. Magic Cap *packages* —the equivalent of programs on a desktop machine—have easy access to e-mail, fax, beaming, printing, and other communication features.

Magic Cap's *ease of use* comes from its graphical interface. Magic Cap embodies the original spirit of the Macintosh—it creates an inviting world that uses familiar objects in common ways. In order to be more than a novelty in the personal electronics world, communicators must be consumer electronics products, not tiny computers. They must be usable by both self-admitted "gadget freaks" as well as people who draw the line at telephones and possibly automated teller machines. This means that communicators must be much easier to use than even the friendliest personal computers. In Magic Cap, ease of use comes from great user interface design with simple, powerful metaphors and consistent features. General Magic has performed an enormous quantity of user testing, and continues to do so, as do its most successful developers.

Magic Cap was implemented with efficiency, utility, and safety as its key goals. *Efficiency* begins with Magic Cap's object-oriented structure. Packages can build on the rich set of objects provided in ROM. *Utility* comes from a runtime environment designed to work with existing development and debugging tools and which assumes that it is running in a very constrained space. *Safety* is embodied by the memory management system: it makes efficient use of limited RAM even when creating and destroying scores of objects, protects user data, and includes a *clean up* feature to recapture memory that a programmer might otherwise leave behind.

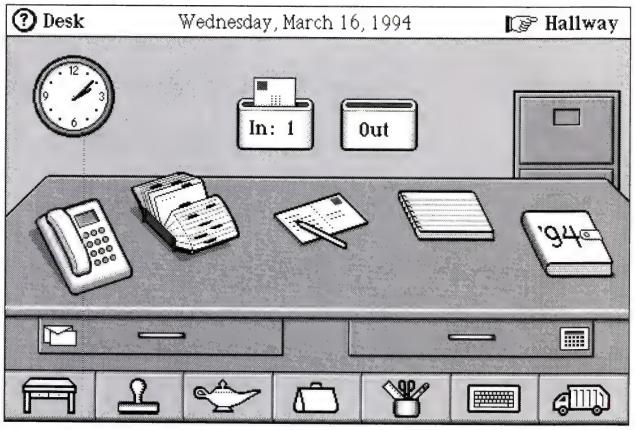
The Magic Cap Hardware Platform

Magic Cap offers another feature that developers can feel good about – a standardized hardware platform. The current generation of communicators is built around the Motorola 68349 *Dragon* microprocessor. The Dragon contains a 68000-family core with memory protection hardware and low-power operating features. Communicators also include at least 1MB of RAM, 3MB of ROM (including the software environment and bundled third-party packages), a data/FAX modem, the *Magicbus* ™ high speed interface bus, a PCMCIA Type II slot, a high-speed infrared transceiver, and an *option* button. Each communicator has a 480 x 320 LCD screen with 4 levels of gray, a speaker, a microphone, and an interface for a headset which turns the communicator into a telephone. These features provide a rich hardware base upon which to build software packages for all Magic Cap communicators.

This minimum hardware requirement ensures that developers can build packages that work on a variety of communicators, yet allow hardware vendors to add unique features. For example, Motorola's Envoy includes a second PCMCIA slot and a wireless modem. These hardware enhancements have been integrated into Magic Cap.

A User's View of Magic Cap

When you turn on a new Magic Cap personal communicator, you'll see the *desk*. This is the jumping off point for most of the things you'll do with a communicator.



Magic Cap's objects suggest items that you use in the real world. To make them work, you touch them: touch your in box to see your mail, touch the datebook to make an appointment, and so on. This display shows several other constants in the Magic Cap world: the upper *name bar* and the lower *control bar*. The name bar supplies help, shows the date and/or time, and contains the *step back* hand to allow for navigation from one Magic Cap package to another. From left to right, the control bar contains:

- A button that returns the user to the desk,
- A *stamper* that contains both decorative and functional objects to drop on other objects (this is also used to access the *magic hat* which allows users to change the fundamental behaviors and appearance of their communicators),
- The *magic lamp* from which you can search, print, fax, mail, and beam (package-specific commands also go here),
- The *tote bag*, like the Macintosh Clipboard, can be used to carry information from one place to another,
- The *tool holder* , which is full of drawing and editing tools,
- A button to display the on-screen keyboard, and
- The *trash truck* to dispose of unwanted or unneeded items.

Though some might dismiss the interface as "cartoon-like," Magic Cap's items are big, clean, and familiar for several important reasons. First, LCD screens on communicators are not the clear bright CRTs that we're used to on computers; small images and fine shadings are harder to see. Second, because many people don't like having to reach for a stylus, Magic Cap

is designed to be used with fingers and fingertips doing the touching (although a stylus also works), so having big objects to touch is vital. Finally, users learn and work better with simple, abstract images than with complex, realistic ones.

This desk scene demonstrates that communication is at the heart of Magic Cap. The center of the screen is occupied by the most important elements of communication: the *in box* (where newly arrived messages appear) and the out *box* where newly created messages go. Surrounding these are other important communicating software packages built into Magic Cap, including a notebook for free-form drawing and writing, a phone that you can use when your communicator has access to a phone line, and a file cabinet for organizing saved messages. The name cards file to the left of center contains postal and e-mail addresses for your associates as well as their telephone numbers.

Magic Cap also applies its communication abilities in more subtle ways. When you use the datebook to schedule a meeting, Magic Cap will offer to create an e-mail message telling the other attendees about the meeting. If the attendees have Magic Cap communicators, they can install the appointment in their datebooks automatically when they get the message, and even generate an RSVP message that comes back to you. If you create a drawing in the notebook, you can mail that page to a colleague's communicator.

These features – an inviting, graphical interface, careful attention to how people use the machine, and a standardized hardware platform – demonstrate Magic Cap's kinship with the Macintosh. The similarities to the Macintosh don't end at the human interface though, as we'll see in the next section.

In Magic Cap, Everything Is An Object

The original Macintosh was also designed with consistency and limited resources in mind. The original Macintosh team's answer provided a "toolbox" in ROM that supplied programmers with many of the common user interface elements. Magic Cap follows this same path by providing all of its object classes in ROM for developers to build on.

Every Magic Cap object descends from class `Object`, thus inheriting the memory management commands, the ability to be encoded and decoded for transmission to another communicator, tools for reading and writing its own data, and debugging code. New classes may also inherit from `Linkable`, which allows you to build lists of arbitrary objects.

Magic Cap builds on the foundation classes to create `Viewable` objects and their subclasses. Anything that gets drawn on the screen and anything that the user can touch is there because of a viewable. Viewable's subclasses include Button, Control, Card, Scene, Form, Color, TextField, and Coupon.

When you look at the Desk scene, you can see several viewables at work. (Each different screen in Magic Cap is known as a *scene*.) The name bar and control bar at the top and bottom of the screen are both viewables, as is the Scene object that lies between them. Each of these viewables contains multiple other viewables. The contained viewables are clipped to their container's boundaries, and receive touches before their container does. We'll see some more examples of viewables when we demonstrate how to build a Magic Cap package later in this article.

Objects, Memory, and Magic Cap

The Magic Cap runtime system has to perform three basic functions: manage objects in memory, execute operations against those objects, and handle any errors that might arise.

Since objects may be created and destroyed quickly within a communicator's limited memory, the runtime system has to work to fill in gaps left by old deleted objects. Thus, just as the Macintosh has *handles* which allow it to rearrange blocks of memory on the fly, Magic Cap has *object ID*s. Each object ID is a 32-bit value that uniquely identifies an object within a package. When you use an object ID, Magic Cap's object runtime quickly and efficiently converts it into a pointer to the object. Because an object ID isn't an actual pointer to an object, the runtime system is free to move the object around in memory to minimize memory fragmentation and thus make room for new objects.

The runtime system also controls how code gets access to data. Magic Cap objects are structures that define both data elements and a set of functions that operate on the data. The data elements are called *fields*, and the functions are called *operations*. Most objects define operations that read and write specific fields, these are called *attributes*. Under most circumstances, an object doesn't access its fields directly but uses either attributes or the special `Field` operation inherited from `Object`.

Since a personal communicator is designed to keep all of its permanent user data in RAM, the runtime system also works to protect this data. Every communicator contains a block of *persistent* RAM which is preserved when the communicator is turned off. (This memory is standard low-power static RAM, so either the main battery, backup battery, or AC power has to be connected.) Persistent RAM is protected from change through a set of internal routines provided by Magic Cap. Packages can't write directly to persistent memory; instead, packages write changes to a buffer area of memory. Magic Cap updates persistent memory from this buffer area periodically, ensuring that packages don't corrupt structures in persistent memory. Because persistent memory is protected from direct change, writing to it is somewhat slower than if memory were not protected, but essential user data is safe.

As a performance enhancement, Magic Cap requires a second kind of RAM that is not protected from change and does not retain its contents when the main power shuts off. This *transient* memory is implemented using low-power dynamic RAM. Transient memory is useful for objects that are created and destroyed within the scope of an existing operation, objects that can be recreated from persistent data, and other temporary objects. If your package uses transient memory, it must be prepared to recreate any objects stored there if the communicator shuts off or is reset.

E.T.O. Now Includes Symantec C++ Version 8.0 for Power Macintosh—A $499 value.

Good News for Macintosh Developers:

Order E.T.O. today and get the Mac™ OS SDK and Apple Developer Mailing for only $99 more (a $450 savings)

All 3 for $1,194
P0175LL/A

Now E.T.O., the deluxe collection of core programming tools on CD, offers more value, convenience and up-to-the-minute technology than ever before. It includes an integrated set of development environments, compilers, debuggers, application frameworks, testing tools, and provides streamlined application development for 68K and Power Macintosh systems. With the versatility of two powerful environments—MPW Pro, Apple's Macintosh Programmer's Workshop, and the popular Symantec C++—you can easily switch between them and use the one that best suits your needs.

An Apple exclusive, E.T.O. is sold as a convenient subscription with new releases and tool updates automatically sent to you. So your tools are kept up-to-date all year long.

Now, for a limited time, you can have all the value and versatility of E.T.O.—plus more. Not only does E.T.O. save you time and money, but you also get an opportunity to order two more exclusive Apple subscription products at a special price. For only $99 more ($1,194), you get E.T.O. with the Mac OS SDK (Software Developer's Kit) and the Apple Developer Mailing, two convenient subscriptions that provide everything you need to move to the top as a first-class developer.

Hurry, offer expires July 31, 1995.

E.T.O.: Essentials • Tools • Objects

Complete First Year Package

$1,095

(M0895LL/C: 1 starter kit plus 2 CD mailings)
Once you subscribe to E.T.O., renewal fees are only $400 per year.

How to Order

To order call:

1 800 282-2732

Or ask for a free copy of the Apple Developer Tools Catalog.
VISA, MasterCard and AMEX accepted.

Mac OS SDK

More than 30 System Extensions in One Convenient Product

$299

(R0603LL/A: 4 CD mailings.)

The Mac OS SDK is a comprehensive assortment of powerful system technologies that will assist you in developing the world class applications of the future. You get convenient and affordable instant access to over 30 popular Macintosh system software extension SDKs that Apple publishes, including QuickTime, AOCE, QuickDraw GX and MacTCP. Mac OS SDK provides all the software and programming information you need to add support for system software technologies in your applications.

Apple Developer Mailing

Important Monthly Industry Updates and Essential Technical Information

$250

(C0197LL/A: 12 issues)

The Apple Developer Mailing provides you with the latest technical information you need to stay up-to-date on Apple programming and at the top of your industry. The mailings include: Apple Directions, Apple's monthly developer business report, and The Developer CD Series, a comprehensive information resource, including on-line technical documentation, system software and programming utilities.

MTM0595

Magic Cap gathers objects into *clusters*. Each cluster is similar to a Macintosh heap; it has a master block that refers to each of the objects in the cluster. Each object ID refers to a master block and contains an index number into that block. Unlike the Macintosh, clusters may cover several disparate area of memory. (In other words, a single cluster might include both the RAM built into a communicator and the RAM on an inserted PCMCIA card.) A package has both a *package persistent* and *package transient* cluster, and may define additional clusters for its own use. Packages also have access to the *system persistent* and *system transient* clusters so they can access system data and call system operations.

The system also maintains a *shadow cluster* for objects in ROM. If the user decides to modify an object in ROM (e.g. changing the background color in the Desk scene), that object is copied into the ROM shadow cluster and modified. The runtime system looks for objects in the shadow clusters before checking ROM. This allows the user to customize any part of the system. This also allows General Magic to patch code in the ROM for software enhancements and upgrades.

A package's clusters are all gathered into the *context* for that package. Only one context may be active at a time. If the user moves from package to package or the preemptive kernel switches execution from one package to another, the context will be switched to that of the current package. The use of contexts and object IDs isolate packages from each other and thus protects both the developer and the user.

Executing Operations

Executing an operation against an object requires both an object ID for that object and the name of the operation to execute. The runtime system will search through the object's class then its superclasses until it finds the appropriate operation and then will execute the code. (Magic Cap draws a distinction between a *method* and an *operation* – a method is the code that implements an operation, thus the `Method` keyword at the start of our code examples.)

Magic Cap packages are written in C with some object extensions, so executing an operation looks just like a standard C function call. The only thing that distinguishes executing an operation from a C function call is the object's ID in the first parameter. This is shown in the code below. In this code, we're first asking the `SimpleAgent` class to create an instance of itself in transient memory. (The trailing "_" in the sample code denotes a class.) This ability is inherited from `Object`. We then extract the text of our message object and copy it into transient memory. Finally, we use an attribute to put our newly copied text into the agent's `message` field and return the agent's object ID to the caller.

```
Method ObjectID
AgentSender_MakeAgent(ObjectID self, ObjectID messageText)
{
  // Create an "agent" object which will carry a message to another
  // user. Return the agent's ObjectID to the caller.
  ObjectID agent, message;

  // Create one object of class SimpleAgent in transient memory
  agent = NewTransient(SimpleAgent_, nil);
  // Read the contents of our message (passed in messageText)
  // and make a duplicate in transient memory.
  message = CopyTextTransient(messageText);
  // Give the copied message to the agent
  SetMessage(agent, message);

  return agent;
}
```

Magic Cap also supports *intrinsic* functions which aren't bound to a particular class and which use a fast dispatching scheme. Certain common system and utility functions are implemented as intrinsics, and are called if they were regular C

functions. (The difference is that Magic Cap locates the intrinsic's code through its dispatching mechanism so intrinsics can be patched later in life. Most C compilers and linkers call functions by their addresses which makes later patching difficult or impossible.)

Error Handling

Since communicators hold important personal information and have no separate mass storage, they must recover gracefully from programming errors without losing data. To simplify error handling and make sure it can detect any errors that a package doesn't, Magic Cap incorporates *exception handling*.

Exception handling isn't new, but if you aren't familiar with it here's the idea: before executing some code that might fail, a routine can add an *exception handler* to a list in the system. When a subsequent routine fails, it creates an *exception* object containing information about the failure and passes it to a system routine which *throws* the exception. Throwing an exception passes control to the last exception handler installed, resetting the stack pointer and the processor's registers along the way. (Receiving an exception is also known as *catching* the exception.) The exception handling routine can look at the specific exception and decide whether to fix the problem or to pass the exception to the next handler in the list.

Magic Cap's exceptions are all subclasses of `Exception` so a package can define its own exceptions. An exception handler may specify that it catches all exceptions or only those of a particular class. Calling `Try()` or `CatchAll()` installs a handler which catches all exceptions. Calling `Catch(`*`exception: Object`*`)` installs a handler which catches a specific class of exceptions. If your code wants to signal an exception, it should call `Fail(`*`exception: Object`*`)`. Exception handlers are removed from the list in two ways: explicitly, or by catching an exception. Calling `Commit()` removes the most recently installed handler.

```
Method ObjectID
MyCardClass_ExceptionDemo(ObjectID self)
  {
  // "volatile" is a C keyword which turns off some optimizations.  In particular,
  // we want to make sure the following two ObjectIDs are allocated on the stack.
  // This is a prerequisite for variables that are used in an exception handler.
  volatile ObjectID formOwnerCard = nilObject;
  volatile ObjectID requestedItem = nilObject;

  ObjectID exc;          // caught exception

  // Push an exception handler on the list, then continue execution.  Pushing the
  // handler returns nilObject, catching an exception brings control back to here and
  // returns a non-nil object ID.
  if ((exc = CatchAll()) == nilObject)
  {
    requestedItem = NewPreferred(MyItemClass_, nil);

    // Set up a form as the "current form".  We want to use exception handling
    // around this so we can reset to the previous "current form" if any subsequent
    // operation fails
    formOwnerCard = BorrowForm(self, true);

    // do something that might fail...

    // Reset the "current form" to its old value
    ReturnForm(self, formOwnerCard, true);
```

```
    // We're at the end of the code that might fail, so remove the exception handler.
    Commit();
  } else {
    // Something went wrong.  If we changed the "current form",
    // restore the old form.  If we allocated memory, dispose of it.
    if (formOwnerCard != nilObject)
      ReturnForm(self, formOwnerCard, true);
    Destroy(requestedItem);
    requestedItem = nilObject;
    // We've handled this exception completely.  If we want
    // another handler to know about this problem, use Fail(exc);
    // to propagate the exception to our caller
  }
  return requestedItem;
}
```

Inter- and Intra-Package Communication

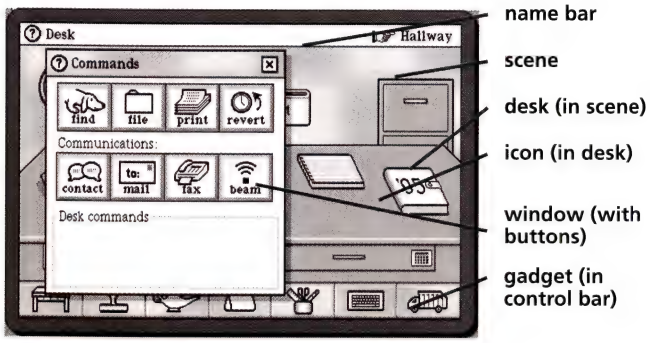
Magic Cap's memory protection services are useful, but what if your package wants to access information in the system or another package? If the system (or a package) places an object ID into a special list, packages can refer to that object by its position in the list instead of using an actual object ID. This alternate form of object ID is called an *indexical*. Indexicals are a powerful tool in Magic Cap – packages can use them to hold package-specific global variables and to export selected objects for other packages to use. Packages can also use *system indexicals* for well-known global objects. The system provides indexical values for nearly everything: images, input devices, text styles, colors, sounds, system error messages, line styles, windows, objects describing the system's current state, and so on. System indexicals deliver a treasure chest of objects for packages.

Packages not only get to read indexicals, but packages also get to replace indexicals with their own object references. Packages can change both Magic Cap's appearance and behaviors in this way. For example, an enhanced Calendar program could replace the Datebook viewable on the desk or the Datebook object itself with a custom version.

Packages can also export code. Developers can release the *class numbers* for classes defined in their packages and the *citation codes* used to identify their packages to Magic Cap. Given these two pieces of information, a developer can instantiate objects using the classes defined in another package while leaving the original package untouched.

Graphics in Magic Cap

Magic Cap's graphics system is built around `Viewable` and its subclasses. Magic Cap already includes viewables for most things that you want to draw: bitmapped images, text, shapes, lines, boxes, arbitrary paths, lists of other items, and so on.



The fundamental viewable in most packages is a `Scene` which contains the package's other viewables as its subviews. Each viewable acts much as a Macintosh `GrafPort` does: it sets up a clipping area within its bounds and supplies a local coordinate system. Each viewable is also responsible for drawing itself when Magic Cap requests it; like the Macintosh, Magic Cap keeps track of which viewables are "dirty" and need redrawing. Magic Cap also buffers the screen image through an intermediate bitmap so that drawing is always flicker-free.

While any viewable can respond to the user's touch, some viewables have been specialized for just this purpose:

- *Icons* go to a destination scene when touched
- *Gadgets* open a target window (which may contain additional viewables)

- *Buttons* execute some code directly
- *SimpleActionButtons* invoke an operation on a target object.

We'll discuss viewables at greater length in a future article.

Other Ways To Program Magic Cap

Not all of the code on a communicator has to be compiled 68349 code. Magic Cap also interprets two additional languages – Magic Script and Telescript. We've already described Telescript as a language for building distributed systems and the language upon which AT&T built PersonaLink Services. Magic Script is a powerful yet easy to use scripting language built into Magic Cap. Magic Script gives developers direct access to selected classes and operations, so developers can use it to control packages without writing C code. In fact, there is a one-to-one correspondence between Magic Script statements and Magic Cap operations.

Building a Package

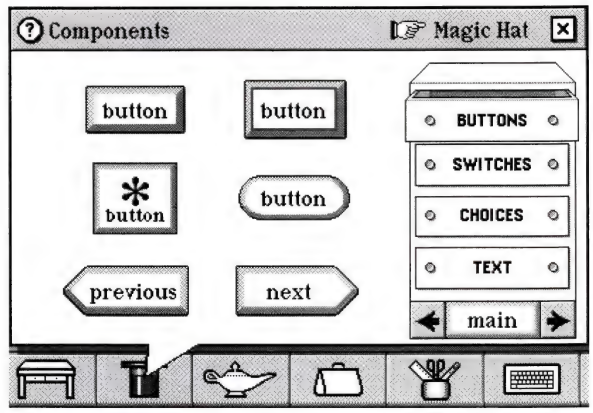
Let's put all of this knowledge to work in constructing a new Magic Cap package. At a minimum, package development involves three files:

- `packageName.Def` which defines the custom classes for a package
- `Objects.Def` which resembles a Macintosh `Rez` source file in that it lists the code and data objects that go into a package.
- `packageName.c` which implements the methods for the classes defined in `packageName.Def`.

These files are built into a package using a standard Macintosh C compiler and linker, and General Magic's custom `ObjectMaker` program. Object Maker performs several functions while building a package: it creates C calling interfaces for the classes in `packageName.Def`, it compiles the descriptions in `Objects.Def` into actual objects, and it merges the compiled and linked C code into the final package. *Magic Developer,* General Magic's fundamental development environment, runs all of these tools within Apple's Macintosh Programmer's Workshop.

Magic Developer is part of the standard Magic Cap Software Development Kit. The development kit includes a special Developer's edition of Magic Cap which runs on any Macintosh with a 68881 FPU (or a Power Macintosh with an emulated FPU.) This kit also includes several sample packages, a code browser, and online documentation.

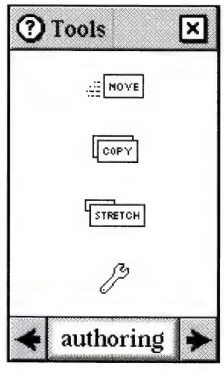
One of the sample packages is `EmptyPackage`: a package which has the name bar, control bar, and an empty scene in between. Many packages begin life as duplicates of EmptyPackage. One such package is `MT-Fortune`, MacTech Magazine's automated psychic advisor. Pressing a button in the package delivers a randomly selected "fortune cookie"-style message to the screen.

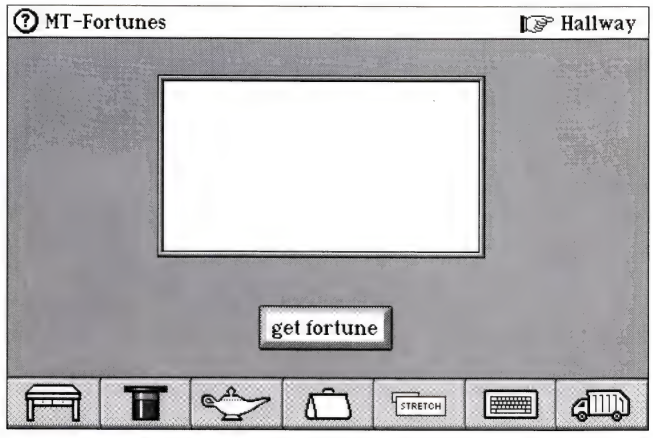


Construct a user interface in Magic Cap

One of the first tasks in developing a new package is laying out the user interface elements. Every copy of Magic Cap can be set to use *construction mode.* This reveals a *magic hat* full of components for customizing the interface or building new packages. Opening the *components* section of the Magic Hat gives us both a button and a text box which we'll place into our scene.

The button and text box provided by the Magic Hat don't fit our needs precisely – the button has the wrong name and the box is too small. We can fix this by touching the tool holder in the control bar and selecting the *Authoring* tools. These tools allow us to move, stretch, and duplicate objects as well as "tinkering" with the appearance and behavior of each object. (The *tinker tool* is represented by a small wrench in the *tools* window.) We can also change nearly any piece of text by creating a *text coupon* from the keyboard and dropping it on the desired object. We used the authoring tools to stretch and center the text box and to change the button's name from "button" to "get fortune."





Write the changes in *Object Maker* format.

Our original `Objects.Def` file doesn't contain these new interface objects, so we have to describe them in the file somehow. The developer's version of Magic Cap can take a package and "dump" all of its objects into a new `Objects.Def` file. We'll do that now, and see that two new objects have been added to the end of the file:

```
Instance Button 'get fortune' 14;
          next: (TextField 15);
      previous: nilObject;
     superview: (Scene 'MT-Fortunes' 10);
       subview: nilObject;
 relativeOrigin: <-2.0,90.0>;
   contentSize: <88.0,21.0>;
     viewFlags: 0x70101200;
    labelStyle: {60,13};     // (TextStyle 8767)
         color: 0xFF000000;
      altColor: 0xFF000000;
        shadow: nilObject;
         sound: {33,11};     // (Sound 'Touch' 289)
         image: nilObject;
        border: {120,19};    // (BorderImage 'But1Up' 213)
End Instance;

Instance TextField 15;
          next: nilObject;
      previous: (Button 'get fortune' 14);
     superview: (Scene 'MT-Fortunes' 10);
       subview: nilObject;
 relativeOrigin: <-5.0,-32.0>;
   contentSize: <236.0,129.0>;
     viewFlags: 0x700D1200;
    labelStyle: {60,13};     // (TextStyle 8767)
         color: 0xFF000000;
      altColor: 0xFF000000;
        shadow: nilObject;
         sound: nilObject;
        border: {120,103};   // (BorderImage 1512)
    fieldFlags: 0x01040000;
     dataStore: nilObject;
     baseStyle: {60,1};      // (TextStyle 8734)
End Instance;
```

Two things become obvious when you first look at these objects: they're subviews of the scene (see the `superview` field), and they're connected in a doubly-linked list. This is typical of viewables.

The `next`, `previous`, and `superview` fields contain typical references to other numbered objects in the same package or references to `nilObject` (Magic Cap's way of saying "this field intentionally left blank.") However, if you look at the button's `sound` field or either `border` field, you'll see an unusual value in curly brackets, e.g., `{60,1}`. These values refer to system indexicals, the system-wide global variables we mentioned earlier.

Create supporting classes and objects.

We now have an interface for our package, but it doesn't do anything special yet. We still need to create a class which can generate and display fortunes. Then we'll attach a Magic Script script to the button to get a fortune from this class and drop the result into the text field. (We could simply subclass `Button` to create and display a fortune when pressed, but that would be less interesting.)

Creating a new class begins in another Object Maker file, `MT-Fortunes.Def` in this case. We'll add the definition for a new class which can generate and return fortunes on demand.

If you have a CD-ROM drive, then *you've gotta have* every article published in the first 9 years of MacTech Magazine!!

FREE UPGRADE TO VOL 1-10!*

... now in THINK Reference format!

Available Now!!

MacTech CD-ROM, Volumes 1-9:

Over 1100 Articles.
All 103 Issues, including all of 1993.
All the Source Code.
THINK Reference 2.0.
Working Applications.
Full Documentation.
Demos for Developers.
And More!!

"When I designed THINK Reference, I envisioned endless databases at my fingertips. MacTech has doubled the information that is just a mouse click away."
– Darrell LeBlanc, Formerly of Symantec, Author, THINK Reference 2.0

"The CD strikes me as an impressive and very useful resource. The only disadvantage I've found is that each answer the databases yield exposes me to so many more issues, that I find myself exploring the articles for the sheer wonder of it, and thus putting off the real coding I should be doing. :-)"
– Nicholas De Mello MacTech CD Beta Tester

MacTech MAGAZINE™ Formerly MacTutor
FOR MACINTOSH PROGRAMMERS & DEVELOPERS

Voice: 310/575-4343 • Fax: 310/575-0925
AppleLink: MT.CUSTSVC
CompuServe: 71333,1063
Internet: custservice@xplain.com
America Online & GEnie: MACTECHMAG

✓ **45 MB of MacTech Magazine articles, Volumes 1-9**
Every article, 1100+ of them, from all 103 issues of MacTech Magazine printed from 1984 through 1993. Articles ranging from Assembly to BASIC, C to Pascal, Forth to FORTRAN and more. *And the articles are in THINK Reference!*

✓ **Hyperlinks to *Inside Macintosh* Databases of THINK Reference**
The articles have hyperlinks to relevant portions of the *Inside Macintosh* databases of THINK Reference. For example, if you are looking for information on Aliases, look at the MacTech articles on Aliases and use the hyperlinks to jump to the *Inside Macintosh* entry for the Alias Manager. And now, with the articles in THINK Reference, you can do free-text searches 8-10 times faster than you could previously with On Location™ 2.0.

✓ **100 MB of MacTech Magazine source code examples, samples, and utilities.**
These are the files that go with the magazine – the code that the articles are talking about. Use them in your own applications, with no royalties!

✓ **A fully-capable version of THINK Reference 2.0.3**
Including *Inside Macintosh* Volumes 1-6 (7 MB).

✓ **Sprocket – MacTech's Tiny Framework.**
Build your simple application using a lean, mean application framework. Experiment with new code without having to build a whole application around it!

✓ **80 MB of FrameWorks, MacApp®, MADA and SFA articles, files and source code.**
The most complete set of FrameWorks archives known.

✓ **Apple APIs, Utilities, and SDKs**
Including: Universal Header Files, Discipline, Macintosh Drag and Drop SDK, MacsBug, Telephone Manager SDK, Thread Manager SDK, TrueEdit 1.8 and and more.

✓ **Ariel Publishing's *Inside BASIC* on disk**
...and other related BASIC programming information and tools.

✓ **75 MB of Special Demos relevant for developers**

FREE!
THINK™ Reference
Symantec's THINK Reference 2.0. Complete on-line guide to Inside Macintosh, Vol. I-VI, with cross referenced index, detailed information of each function, procedure and detail needed when programming the Macintosh.
SYMANTEC.™

MacTech CD-ROM, Volumes 1-9:
$199 plus shipping and handling. $69 plus shipping and handling for upgrades from any previous version of the CD

* All purchasers of the 1-9 CD will receive a FREE upgrade to the next version of the CD when in becomes available.

```
Define Class FortuneMaker;
  inherits from Object;

// Two data values: a block of characters that contains all of our fortunes (separated by
// newlines) and a number which refers to the fortune currently being displayed.
  field fortunes:    Text, getter, setter;
  field fortuneNum:  Unsigned;

// An attribute used for reading and writing the "fortunes" field above.
  attribute Fortunes: Text;

// A bit of code to select and return a fortune.  The options after "Text" make this
// operation visible to Magic Script.
  operation GetFortune(): Text, safe, common, scriptable;
End Class;
```

This class includes a long text object which contains all of the possible fortunes, an integer which stores an ID number for the last fortune returned, and attributes to read and write the text string. This class also defines an operation to generate and return a new fortune string. The suffix `safe, common, scriptable` makes this operation available to Magic Script so we can demonstrate scripting later.

Now that we have a template for the class, we must write its `GetFortune` operation. `MT-Fortunes.c` contains this code:

```
Method ObjectID
FortuneMaker_GetFortune(ObjectID self)
{
  ObjectID newFortune = nilObject;
  ObjectID ourFortuneText;
  TextRange  selectedLine;
  short    numLines;
  unsigned newFortuneNum;

  // Get a reference to our "fortune" field, which is a Text object,
  // then count the lines in this field
  ourFortuneText = Fortunes(self);
  numLines = Lines(ourFortuneText);

  // Pick a new fortune randomly, checking to make sure that it
  // isn't the same fortune we're already displaying
  for (;;) {
    newFortuneNum = ShortRandom() % numLines + 1;
    // Read our "fortuneNum" field without using an attribute
    if (newFortuneNum != Field(self, fortuneNum)) {
      // Store the new value
      SetField(self, fortuneNum, newFortuneNum);
      break;
    }
  };

  // Select a random line in this field and build a text
  // object in transient memory containing the selected text.
  LineToTextRange(ourFortuneText, newFortuneNum, false,
          &selectedLine);
  newFortune = CopyTextRangeTransient(ourFortuneText,
              &selectedLine);

  return newFortune;
}
```

This code reveals several facts about Magic Cap programming. First, note that the object references are all untyped object IDs. Magic Cap defers object type checking until runtime. Next, we're using `Fortunes()` which is a *getter attribute* that reads our `fortunes` field. We requested

one of these by appending `getter` to the field in `MT-Fortunes.Def`. We didn't request an attribute for `fortuneNum`, so we're using the built-in functions `Field()` and `SetField()` to read and write this field. Finally, we're returning our result by creating a new `Text` object in transient memory. Since we'll take this object and copy its contents elsewhere, we didn't need to allocate it in persistent RAM.

We now have the definition and code for a class, but our package doesn't actually contain any objects of this class. We need to go back to `Objects.Def` and add a `FortuneMaker` object:

```
Instance FortuneMaker 17;
     fortunes: (Text 'fortunes' 18);
   fortuneNum: 0;
End Instance;

Instance Text 'fortunes' 18;
        text: 'You will meet a tall, dark stranger\n';
End Instance;
```

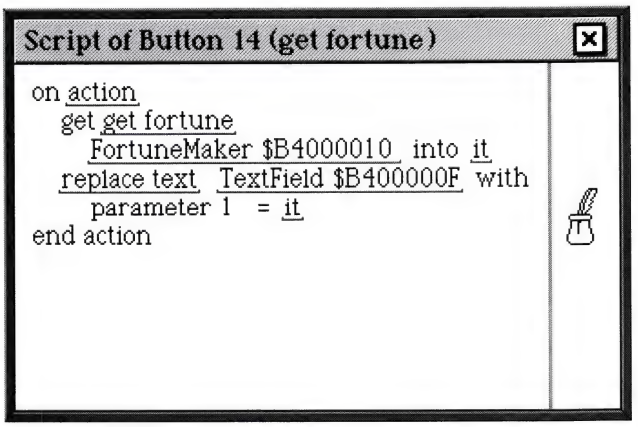
The above lines add our object to the package, but the package doesn't have any way to refer to them. If a package needs to refer to specific objects, most programmers create a list of references to those objects and place them into one of the *package indexicals*. Just as a system indexical represents an object available from anywhere in Magic Cap, a package indexical represents an object available from anywhere in the package. We'll put our FortuneMaker object into a list then install it into one of our software package's indexical lists.

```
Instance ObjectList 'Indexicals' 16;
         length: 1;
         entry1: (FortuneMaker 17);
End Instance;

Instance SoftwarePackage 'MT-Fortunes' 1;
         length: 32;
    dateCreated: 49771;
    timeCreated: 0xFFFFFFFF;
   dateModified: 49771;
   timeModified: 0x041B65B0;
         author: {141,1}; // (ProviderAddressCard 874)
     installList: (ObjectList 'Install' 2);
      receivers: (ObjectList 'Receiver' 3);
   installFlags: nilObject;
installParameters: nilObject;
 installTargets: nilObject;
   autoActivate: true;
       citation: (Citation 4);
      publisher: {141,1}; // (ProviderAddressCard 874)
persistentShadowSize: 0;
persistentChangesSize: 0;
  transientSize: 0;
gotoActionSelector: 3.s;
         hidden: false;
   dontSaveData: false;
 copyOnActivate: false;
 dontDeactivate: false;
     needsReset: false;
systemPackageReserved6: false;
     // skipped a range of reserved fields, all of which are nilObject
systemPackageReserved16: false;
         entry1: nilObject;
     // skipped more empty fields
         entry6: (ObjectList 'Standard Places' 5);
     // skipped more empty fields
         entry9: (ObjectList 'Objects With Help' 6);
        entry10: (ObjectList 'Help On Objects' 7);
     // skipped more empty fields
        entry13: {40,7}; // (Scene 'Hallway' 88)
     // skipped more empty fields
        entry25: (ObjectList 'Indexicals' 16);
     // skipped more empty fields
End Instance;
```

Note: *We didn't have to set up any of the fields of this object except for the indexical we installed. The values were copied from EmptyPackage, the sample used to start the MT-Fortune package.*

Add the script

Since we want to tie the FortuneMaker object to the button with a script, it's time to compile the package and run it. Once we're in the package, we can select the tinker tool, hold down the option key, and touch the button to edit its script.



Magic Script is extremely easy to use. Touching the ink well at the right side of the window inserts a new statement into the script, with the entries for the command and the *responder* (the object which will execute the operation corresponding with this command) underlined. You can touch an underlined object to change it, or touch an underlined field to get a pop-up menu of all possible commands that go there. While a full discussion of Magic Script is beyond the scope of this article, you can get all of the details from the documentation in the software development kit.

Once we've added a script to the button, our package is nearly done. The package should be dumped back into `Objects.Def` so we can build future versions which contains the script. `Objects.Def` also contains information about the package's authors and some help text; these should be updated. Still, you've seen enough that you should be able to begin experimenting with the Magic Cap software development kit as soon as you get one.

Starting Down The Road To Magic Cap

Now that you've taken a look inside Magic Cap, you may be wondering how you can bring your software to this powerful new platform. General Magic has teamed up with Metrowerks to release Magic Cap software development tools by the summer of 1995. You can receive information about General Magic's developer program and a notice when the development kits become available by sending electronic mail to dev-info@genmagic.com or by calling (408) 774-4000.

By Jon Wiederspan, jonwd@tjp.washington.edu

Your Very Own Web Server - MacHTTP

Take Five Minutes and Join the Web

[Remember that excitement the first time you ever saw a Macintosh? How about that first week after you got your hands on one? I don't – I simply dove in and resurfaced days later, grinning from ear to ear. From time to time, we come across technology tools which evoke such reactions. Serving up pages on the World Wide Web seems to have this effect on some people.

*In this article, Jon introduces us to MacHTTP, a Macintosh server. It couldn't be easier, nor could it be any more addictive. In addition to job-security warnings, and tips on getting along with un*x admins, he also gives us a preview of the 3.0 version (coming soon), and shares tips on how to get the most out of your Web site. You don't have one yet? Odds are good that, after reading this article, you will! Enjoy! Let us hear from you about your experiences, and send us your URLs – Ed stb, editorial@xplain.com]*

Like most people on the Internet, I have only recently been introduced to the World Wide Web. It was less than two years ago, that fateful day when I downloaded a copy of Mosaic from the UMich ftp site. One double-click and I was launched into hypermedia heaven, madly clicking my way around the globe, heedless of restrictions like server addresses and directory paths that I had accepted without question for years in my FTP client. The memory already fades, so I don't recall exactly whether I rushed right over to show it to my friends or whether I waited a few minutes first to get a drink of water. Either way, though, it was not long before we were engrossed with ideas of what *we* could do with our own World Wide Web server.

That is when reality slammed me rudely back to earth. Yes, I had a Sun workstation and the server software (NCSA httpd was my choice) was freely available, but my novice UNIX skills had been barely sufficient to get an anonymous FTP site started and building and running the httpd software turned out to be completely beyond my capabilities. I didn't have the time to spend improving my skills and I couldn't justify the funds to hire someone who could do the job. It looked as if I was to be denied my part in the fastest growing service on the Internet.

Luckily for me, there was a solution in the form of MacHTTP, a software package from BIAP Systems in Houston, Texas. MacHTTP is a high-performance, feature-packed WWW server that runs right on your own Macintosh under the standard Macintosh OS. In this article I will show you how to install MacHTTP, how to fine tune its performance, and how to use some of its special features. I will also talk about why anyone would want to run a server in the first place and some issues to consider in site design and maintenance.

In order to keep everyone on the same page, there is some terminology everyone should know. If you aren't completely familiar with terms like WWW, URL, FTP, and HTTP, you should read the sidebar entitled "Vocabulary 301". They will keep

Jon Wiederspan – Jon runs a WWW site for the Technical Japanese Program at the University of Washington. In addition to valuable information about the program, he also provides information useful for building a WWW server using MacHTTP or running other TCP/IP services on a Macintosh. You can check out his pages at http://www.uwtc.washington.edu/JonWiederspan/JonBio.html or send him e-mail at "jonwd@tjp.washington.edu".

VOCABULARY 301

Here are terms you need to know to when trying to set up a World Wide Web site (or read this article).

FTP – File Transfer Protocol. A very common and basic method of providing files for sharing with others.

Gopher – A more efficient type of FTP which allows more users to connect at once. Also provides a way for one Gopher server to provide links to other Gopher servers.

HTTP – HyperText Transfer Protocol. A way of communicating that is based on Hypertext files which contain links to other files.

NNTP – Network News Transfer Protocol. A way of transferring news messages around on the Internet.

SMTP – Simple Mail Transfer Protocol. A way of transferring mail around on the Internet.

Telnet – One of the earliest network services available. It allows you to connect to a remote computer and issue commands as if you were physically at that computer.

URL – Uniform Resource Locator. A way of specifying exactly where a file is on the Internet and how to connect to it.

WWW – World Wide Web. A virtual network of servers which includes HTTP, FTP, Gopher, and other servers connected by URL's.

popping up throughout this article and I'm not going to take time to explain them.

BEFORE WE BEGIN

Before we discuss how to start your own World Wide Web server we need to cover some other topics.

Why Run a Server

The first topic is, "*Why do you want to start your own server?*" I warn you right now that a World Wide Web server is a time sink that will gradually steal away more and more of your working day until you have time for nothing else but building your site. This is not because maintenance is so difficult. In fact, maintenance is an almost negligible task for most sites (but don't tell your boss that). The truth is that there are so many cool things you can do with a Web site that its hard to resist adding just one more hack and each hack takes a little bit longer than the last. So, if you plan to run a site, make sure you have the willpower to resist this temptation (or have a very secure job).

Now that I am legally covered, we can discuss how you could benefit from running a World Wide Web server. Here is a list of good reasons that I keep handy:

- To advertise products or services for your company or yourself
- To attract potential customers for your company, organization, or non-profit group
- To publish research results
- To provide online help resources for customers
- To publish informational articles, newsletters, magazines, or anything else
- To provide an easier front-end to existing network services

like FTP or Gopher sites
- To provide easy access to databases of all kinds
- To conduct surveys or gather opinions or comments on any topic
- To make personal information like your photo, resume, or opinions available to the general public.
- To put up pictures of your last vacation or your pet dog (cats are acceptable as well).

As you can see, there are a number of ways that a World Wide Web site can be useful.

Is putting up a WWW site a good idea for you or your company? If your company already offers services on the Internet (mailing lists, FTP sites, WAIS databases), then the decision is fairly easy. A WWW server can provide a much nicer interface to all of these and even help unify multiple services for your customers. However, if this is your first foray into Internet services you need to give this some thought. Think of the kind of people that inhabit the Internet. Now think of your product or service. Do they want it? If not, then this probably isn't for you. But don't let that stop you...

Uniform Resource Locators

The next topic to cover is Uniform Resource Locators or URL's. URL's are text strings that completely specify where a file or directory can be found on the Internet or a local computer. They form the connective filaments of the World Wide Web, connecting servers which run a variety of protocols from HTTP to Telnet.

A URL uses the following format:

protocol://server_address[:server_port]/[directory/][filename]

protocol can be any of http, ftp, gopher, telnet, nntp, mailto, or any other communications protocol that works on the network. This tells the client software how to communicate with the server. It is up to the client software to support the protocol, as in "mailto", which is not supported by all clients.

server_address is either the IP address or DNS name of the server which has the file.

server_port is an optional number which designates what port the server communicates on. We'll talk more about that later, but if nothing is specified, the standard port of 80 is used by default.

/ indicates the root directory for the site. Only files in this directory tree can be accessed.

directory/ is a sub-directory or directory path that contains the file. "files/reports/daily/", for example.

filename is an optional string telling which file should be returned. If no filename is specified, then a default action is done for the folder. In some cases (depending on the server) a listing of the directory is returned. This is not very safe, as it gives clients access to every single file on your site. MacHTTP uses a different method that will be discussed below.

In summary, the URL specifies what method to use to connect to the server, what server to connect to, what directory to look in, and what file to return in that directory. Since all of this information is provided in the link, the user doesn't need to know anything about such things and can click on any link

without worrying about where it leads. This is a simplification, of course. More information about URL's can be found at http://www.w3.org/hypertext/WWW/Addressing/Addressing.html

Choosing a Server

The final topic to cover is that of selecting the proper server. If you have to deal with UNIX-loving network administrators, then you will need rational reasons for selecting a Macintosh as your server. Use the following guidelines to help you make your decision.

Performance – I'll talk more about this later, but suffice it to say here that a Macintosh server running MacHTTP will outperform an equivalently priced UNIX workstation running either the NCSA or CERN servers.

Installation – This is the real Macintosh strength. Even a fairly inexperienced Macintosh user can get MacHTTP running in very little time. On the other hand, even a very experienced UNIX person may have some problems installing the NCSA or CERN servers. This stems primarily from the fact that MacHTTP comes as a fat binary which is ready to run on any Macintosh, while the NCSA and CERN servers need to be compiled and there is a tremendous variety across the platforms they support.

Server Maintenance – MacHTTP 3.0 can be completely monitored and controlled remotely with an easy-to-understand graphic client. I have yet to see another server that offers this ease of maintenance.

Server Stability – Don't believe the rumors that Macintoshes crash more than other systems. If you have a lot of

freeware system extensions installed (as I do) then that might be true. Without those extensions, though, your Macintosh will keep running for months or even years without crashing.

OS Design – Another vicious rumor says that the Macintosh can't be a server because it doesn't have pre-emptive multi-tasking. Actually, this is not necessary because the server software itself provides the multi-tasking using the Thread Manager.

Site Content Maintenance – If the people who will be adding content to the site are also Macintosh users, then this is a major consideration. By running a Macintosh server, they can mount the files directly on their computers for easy editing.

Starting A Site

So now you're ready to begin. There are really only three things you need to start your WWW site: a Macintosh computer, the MacHTTP software, and network connection (not strictly required). The Macintosh must be running MacOS System 7.1 or later, but any Macintosh from a Plus on up can be a server. Either MacTCP or OpenTransport (if it is released by the time this is printed) are needed for network connectivity and a minimum of 600K of free memory is required to run MacHTTP. If you want people to be able to reach your server, you need a dedicated connection to the Internet and an IP address. The documentation that comes with MacHTTP includes instructions for running on a standalone machine, though, if you need to do so for demonstration or development purposes.

See the end of the article for details on where to get the software.

Installation

The MacHTTP software distribution comes with a complete WWW site in a server folder that you can copy onto your hard disk. It doesn't matter where on your disk you copy it or what name you give the folder. This is because once you launch MacHTTP, whatever folder it is in becomes the root of your WWW site. The URL's you use on the site will all use paths relative to this root so it doesn't matter to them whether your server folder is at the root of the hard disk or buried 10 folders deep.

MacHTTP is preconfigured with settings that are appropriate for the majority of sites. This means that installation amounts to four steps:

1) Copy the server folder over to your hard disk.
2) Give the folder a name that you like.
3) Open the server folder
4) Double-click on MacHTTP

Congratulations! You now have a fully-functional site. That's all there is to it. You can access your new server from your favorite WWW client by using the following URL: "http://your_ip_address/"

where "your_ip_address" is the IP address the machine you are running MacHTTP on (e.g., 129.45.3.100).

About MacHTTP

Now that you have MacHTTP running, it is probably time to learn a little more about it. MacHTTP is the product of Chuck Shotton of BIAP Systems, Inc. It was the first HTTP server available for the Macintosh and now offers all of the features you expect of professional HTTP server software including:

- multiple, threaded connections *
- HTTP 1.0 support (also an HTTP 0.9 compatibility mode)
- CGI application support
- site- and document-based security
- secure connections ‡
- remote site monitoring and administration
- gateway for credit card payments (First Virtual)
- support for database and text-searching systems (WAIS, AppleSearch, Verity)

* MacTCP limits to 48 connections
‡ due Q2 1995

MacHTTP also offers several special features that make it easier to use than other servers, especially in a Macintosh-based installation:

CGI (Common Gateway Interface)

The Common Gateway Interface is a proposed standard for information services to communicate with external applications. So far, only HTTP servers take advantage of this standard, and MacHTTP follows the latest CGI standards. The advantage lies in the fact that MacHTTP uses AppleEvents to communicate with CGI applications and those applications can, in turn, use AppleEvents to communicate with other Macintosh applications. This means that WWW pages on your Macintosh can be used as a front-end to databases, spreadsheets, word processors, graphics applications, and anything else you can think of. Pre-built CGI applications are already available for doing maps, linking to FileMaker and Butler databases, and many other specialized tasks. You can also write your own CGI applications using almost any language, including C, Pascal, Prograph, or even AppleScript or HyperCard.

MacHTTP Manager

Beginning with version 3.0, MacHTTP will become a server with very little interface. All management, monitoring, and configuring will take place through the MacHTTP Manager. The Manager can communicate with the server via AppleTalk or TCP/IP and can communicate with multiple servers at one time. This has several advantages that are not found with other servers.

- **Update server settings remotely**. Multiple servers can have their settings updated simultaneously. For large MacHTTP sites which are running multiple, mirrored servers, this is a great asset in keeping them in synch.
- **Monitor multiple servers**. Log data from multiple servers can be displayed on one remote machine for easy monitoring.
- **Security**. Servers can be run "headless" to prevent

unauthorized access (and save on monitors). All other security settings for site and file access can be set remotely and can be shared across multiple servers.

Pre-Processing

MacHTTP 3.0 allows you to use CGI applications to process files **before** they are passed to the client. This could be used to implement server-side includes, on-the-fly document translation, or your own security scheme. There will also be new definable types in addition to the built in TEXT, CGI, and SCRIPT types. MacHTTP will match files to these types based on filename extensions and pass all files of one type to a specified application for pre-processing. As an example, all files with the extensions ".sit" might be passed to a pre-processor for binhex encoding before sending them to the client.

Configurable Log Files

Beginning with version 3.0, MacHTTP log files can be configured to store only the data needed for each site. The site manager can select both which items are logged and what order they are written on the line. The items include date, time, referer, client address, requested file, data transfer speed, and HTTP code.

The log information can be directed to external applications for processing (including to a MacHTTP Manager). This means that connection data can be dumped to your favorite database to track connection statistics or generate billing data for clients.

Aliasing

Version 3.0 of MacHTTP supports using aliases of folders or disks to extend your site. Place the alias inside the MacHTTP folder and you can then write links to files inside the folder or disk. This is very useful when your site outgrows the current disk or for helping multiple groups and users publish information on a single server.

Special Files

MacHTTP supports three special files: Error, Index, and NoAccess. The **Error** file is returned whenever the server is sent a URL that it can't resolve to a file or folder. The Error file can be anything from a simple HTML document informing the user that an error occurred to an advanced CGI that tries to figure out what the URL was meant to point to. The **Index** file is returned whenever (1) a URL is received that specifies only a folder, not a file and (2) a file with the name given for Index files is found in that folder. If no Index file is found in the folder, then an error is returned to the user. Like the Error file, the Index file can be anything, including an HTML file listing the folder contents (or just the ones you want others to know about) or a CGI that allows users to search for contents. In addition, each folder can have its own Index file or no file at all so every folder need not be treated the same. The **NoAccess** file is returned when access to the a file is refused because the user lacks permission to access it. This file also can be anything the administrator wants

and can be used to let the user know exactly why access was refused and what to do if there was a problem.

MacHTTP Performance

Whether you're planning a large site for a corporation or a small site to list the results of your kid's softball team, one thing is certain: you want the site to be fast. Internet users are getting spoiled by near-instantaneous access to sites around the world so many of them won't hang around if it takes too long to download one of your pages. So far, though, there has been no metric discovered that measures how "fast" a server is. Some people look at the number of simultaneous connections that a server can handle, but that is an increasingly useless statistic due to speed increases in both the server software and CPU's. Consider the fact that a server that processes every connection in at least three seconds (not unreasonable) can handle more than 200,000 connections every week *and never process more than one connection at a time.*

The real key to speed on a server is simply **get the data out fast**. Toward that end there are several things you can do to improve the speed of your MacHTTP server.

Server Settings

The easiest way to improve server speed is to adjust the settings listed below:

Thread Manager – This is the single best performance improvement you can make to your site! Versions of MacHTTP from 2.0.2 on are able to use the Thread Manager, if it is installed. The Thread Manager is a system extension that allows applications to run multiple simultaneous threads. MacHTTP uses this to provide each connection with its own thread. Without Thread Manager, connections fight for attention in MacHTTP and slow connections eat up more time than fast ones, which slows your server down. With threading, each connection is somewhat isolated from the others so fast connections are processed quickly without waiting for slower ones. The Thread Manager extension is distributed with MacHTTP and is also part of System 7.5.

DNS Name lookups – MacHTTP logs the IP address of each connection. It also offers the option of logging the DNS name instead. This can be a lot of fun if you're the kind of person who would stare at the log for hours waiting for an "important" site to connect ("Look, someone from Apple just looked at my bio page!"). This is a significant performance hit, though, because every connection adds a delay while MacHTTP waits for a DNS server to return the DNS name. In addition, for sites that don't have DNS names you have to wait for the DNS server to give up looking for a name which takes much longer. For best performance, turn DNS lookups off.

Dump_buf_size – This is the setting which controls how large a block of data is sent to the client at once. The larger you make the blocks, the faster the data can be pumped out. The size option ranges from 512 bytes to 10K. Your best performance will come by using the 10K setting *if* you have the

Thread Manager installed. If you don't have Thread Manager (see above), then the 10K setting may actually slow down your machine when you have to deal with slow clients. There is another possible problem with having a large `dump_buf_size` setting. This problem is mainly with Windows clients and arises when the TCP/IP interface that the client uses is unable to properly handle large data chunks. If the interface tries to reassemble the entire 10K block before passing it on to the client and if the interface has an 8K or smaller limit on how much data it can handle then the interface can crash. If you get a lot of complaints from Windows users, try reducing `dump_buf_size` to 8K or less.

Timeouts – MacHTTP allows you to set how long it will wait before timing out a connection. A timeout occurs when MacHTTP fails to get a response from a client in the time set in the Timeouts setting. MacHTTP has no way to know if the connection to the client was lost or is just slow, so it just hangs on as long as it can. Longer timeout settings will result in extra connection processing as MacHTTP keeps listening for a response from these dead connections. However, using a very short timeout setting can cause slow clients or those with poor network connections to be cut off prematurely. In addition, the Timeouts setting controls how long MacHTTP will wait for a response from a CGI application before giving up. If you are running CGI applications that do a lot of processing, you will probably want to use a longer Timeouts setting to give the CGI time to finish. Otherwise, I recommend a shorter setting for better performance.

Foreground Operation – For best performance, keep MacHTTP running in the foreground. Because the Macintosh uses cooperative multi-tasking, the foreground application is in charge of deciding when other applications get to run. Keeping MacHTTP in the foreground (meaning that its menubar is the one showing) gives it control of the CPU and the most processing cycles.

Hardware Improvements

This is where the largest performance improvements come from, but its not cheap or even possible in some cases.

Network – It is likely that the single biggest improvement you can make is in getting a better network connection. Even a Quadra 610 can swamp most networks, so it's no surprise that relatively slow machine can swamp a 56K Internet connection. This is because the HTTP protocol is not processor-intensive – most of what MacHTTP does is pump data out over the network. Since your network speed comes nowhere near that of your disk drive interface, the network is the most likely bottleneck.

Computer platform – While CPU speed isn't the largest factor in performance, it does make a difference. My general feeling is that the difference between CPU's is much more pronounced than clock differences on the same CPU. That means that you will see a much bigger difference moving from a 68030 to 68040 or from 68K to PowerPC than you will moving from a 66MHz PPC to a 100MHz PPC. In addition, you can get much greater improvements by running two 6100's in parallel than using one 8100 for about the same price (see below on

RAIC Design). Remember, though, that all of this is limited by your network. Its not much good moving from a PowerMac 6100 to a 9150 for speed improvements if you're running on a 56K connection. Using my own site as an example, a Power Macintosh 6100/60 (with Thread Manager installed) can easily handle 20,000 connections a day. It can handle even more than that if you're not running other software (mail, ftp, word processing) on the same machine and have a good network connection. For working examples of MacHTTP on various CPU's, check out the MacHTTP Registry (see this month's "Universal Resource Locator" for the URL).

Disk Drives – Adding a faster disk drive can provide some performance improvement, especially if you are serving a lot of large files or are using an older Macintosh with the original 80MB disk. The difference isn't that great, though, so I suggest adding a new disk only if you need the storage space.

Memory – Adding more memory will have almost no effect on performance. More memory is required to handle more connections, but you can handle the maximum number of connections for your Macintosh in 4MB of RAM, so any 8MB machine already has plenty of memory. If you are running a lot of CGI applications or linking to external applications, then you may need more memory to handle those (about 32MB extra if you're linking to Excel).

RAIC Design

If you plan to run only a small or moderately busy site, then you can probably skip this section. If you plan to put up something that can handle 100,000 connections a day, though, you need to consider using a RAIC design. RAIC ("Redundant Array of Inexpensive Computers") is my obvious play on RAID, which has become a popular method of designing large, fast hard disks. A RAIC system uses multiple Macintosh computers (at least all of mine do) to share processing duties for a site, thus providing much better performance than could be achieved by any single machine purchased for the same price. There are a number of ways to divide processing among several servers and they can be mixed and matched as needed.

Server Mirroring – The latest versions of BIND allow you to have a single DNS name map to multiple IP addresses with connections being passed out in round-robin fashion to each machine in turn. This allows a cluster of computers to appear to be one computer to the outside world. This has obvious benefits in server design. By mirroring the contents of a WWW site across multiple servers you spread the connection load across all of the machines evenly so no single machine has to handle a very high load. This allows several cheaper machines to provide the performance of one more expensive machine. As an example, 10 Macintosh SE/30 computers, each capable of handling 10 simultaneous connections easily, can be combined to create a virtual server that can handle 100 simultaneous connections without coughing and up to 480 simultaneously before dying. All at less than the cost of a high-end workstation. Performance is not the only reason to go with a

RAIC design, though. There is the additional benefit of being able to **hot-swap the CPU's** that make up your server. Since the DNS server takes care of connections, any single CPU can go down or be removed (for maintenance, update, or replacement) without shutting down the site. This gives your site a degree of fault-tolerance that can't be matched by a single CPU. In addition, there is no need for all of the CPU's to be identical. Any Macs that you have lying around and gathering dust can be used to create a RAIC server.

Server Distribution – Another way to improve performance with multiple CPU's is to divide your site up into discrete units that can each run on a separate machine. This is very useful for sections that have high traffic and cannot be easily mirrored, such as a Comments page where people use a form to leave their opinions about your site. This can also be used to give users the illusion of a faster site. A user's perception of your site speed is developed largely on the first connection to the site. By moving the site's home page or a single, very popular page to it's own server, the user gets a very fast initial response and that impression will last even if other pages are not delivered quite so quickly.

Element Distribution – You can also speed up performance by spreading out the elements of your site, by which I mean the HTML pages, graphics, and other large files. Even an older Macintosh can serve up smaller (<10K) HTML pages with great speed. Graphics and other large files take a bit longer, though. In addition, newer clients like NetScape Navigator use multiple connections to retrieve the graphics in a page at the same time as the text, thus placing a larger load on the server. By putting graphics onto a separate server, clients may receive your pages faster. Moving large files to a separate server will also prevent slow connections from tying up connections on the main server.

Application Isolation – The final method to consider is moving external applications and/or CGI applications to a separate server. A CGI application must reside on the same machine as any HTML page that links to it.

You may not want to run another server simply to handle a single CGI application. However, you may want to set up a second machine to run applications (e.g. FileMaker) which CGI applications call. This second machine doesn't need to run a copy of MacHTTP.

As I mentioned above, if you are running a CGI that does database queries or that captures real-time images, or anything else that is processor intensive, you can greatly speed up both the CGI processing and MacHTTP performance by moving the external application to its own CPU.

Keep these techniques in mind the next time you wonder what to do with those older Macintoshes, or when some UNIX user spouts off that you need an SGI workstation to get a really fast site.

Security

MacHTTP provides several options for security on your site. These are in addition to the security that every Macintosh enjoys

by having an operating system that can't be hacked into the way many UNIX systems can. There is no way for someone to launch software on your server, or erase the disk, or change your WWW files (unless you specifically write some software to do this, in which case you deserve what you get).

Directory Restrictions – The first way that MacHTTP provides security is by limiting connections to sub-directories under MacHTTP. Any file that is not in the MacHTTP sub-directories cannot be transferred unless the site manager makes an alias to it. In addition, MacHTTP does not allow people to get directory listings as some servers do (you *can* allow this if you want to, but it isn't built into MacHTTP). This means that people only see files that you provide links to, unless they are good at guessing file names. In addition, MacHTTP does not provide directory listings when the URL specifies only a directory and not a file. Many people consider this a feature on UNIX servers, but it is actually a breach of your site security. If, by some chance, you do want to offer this capability, you can do so by using a CGI application and you can limit it to specific directories.

Allow/Deny – MacHTTP provides the ability to allow or restrict access to your entire site based on either IP addresses or DNS names. It can handle partial addresses and names as well and the rules can be layered so that a single element is allowed within a larger group that is denied. Sites that fall in the "DENY" listing of addresses will be returned the "NoAccess" page. There is no way for users to hack around this protection.

Realms – Realms are used to restrict access to specific portions of your site or specific pages. A Realm is a text string. If this text string is found in the name or path of the file being requested, then the client is asked to provide a username and password to access the page. Each Realm has a username and password assigned to it. If the user cannot provide the username and password then the "NoAccess" page is returned instead.

You're On Your Own Now

Well, that's it. You now have enough information to start your own WWW site on a Macintosh computer. I have completely ignored some important issues, of course, such as network connections, design issues for the content of your site, client software, and the future of the Web. There just isn't room to cover all of those topics, even if I took up the whole magazine. For more information you can check out the links provided in this month's "Universal Resource Locator", or read the "comp.www.*" newsgroups, or watch your favorite bookstore for one of the dozens of books that are sure to appear on the topic.

Obtaining MacHTTP

MacHTTP is available from BIAP Systems, Inc., 16323 Hazy Pines Ct., Houston, TX 77059. On the Web, http://www.biap.com/, by e-mail info@biap.com.

UNIFORM RESOURCE LOCATORS

By John Kawakami and Scott T Boyd

Thanks to Heath Horton, Hazem Sayed, and Jim Straus.
Spare your fingers and find this list online at:
http://www.class.com/MacTech/URLs.html

Internet-related Material

Bolo	http://bolo.ncsa.uiuc.edu/
Consensus	http://www.consensus.com:8300
CU-SeeMe	**http://www.jungle.com/msattler/sci-tech/comp/CU-SeeMe/**
DigiCash	**http://www.digicash.com/ecash/ecash-home.html**
Info-Mac Searcher	http://www.mid.net/INFO-MAC
Internet Config	**ftp://ftp.share.com/pub/internet-configuration/**
InterNIC	http://www.internic.net/
ISDN page	http://alumni.caltech.edu/~dank/isdn/
Peter Lewis	ftp://amug.org/pub/peterlewis
MacHTTP (see also WWW)	
Mailing List	**http://www.biap.com/machttp/mailing_list.html**
Registry	**http://www.batnet.com/ape/machttp_talk/machttpservers.by.mac.html**
Extending MacHTTP	http://www.uwtc.washington.edu/Computing/WWW/ExtendingMacHTTP.html
MacWeb	**http://galaxy.einet.net/EINet/MacWeb/MacWebHome.html**
Matthias Neeracher	http://err.ethz.ch/members/neeri.html
OpenTransport/TCP	**gopher://seeding.apple.com**
Outland	ftp://ftp.outland.com/
Portable Net. Graphics	**http://sunsite.unc.edu/~boutell/png.html**
Eric Scouten (TCP)	http://tampico.cso.uiuc.edu/~scouten/
SGML, Info	**http://www.sil.org/sgml/sgml.html**
C Parser	**ftp://ftp.jclark.com/pub/sgmls**
C++ Parser	**ftp://ftp.jclark.com/pub/sp/**
WWW, Creating a Site	**http://www.uwtc.washington.edu/Computing/WWW/Macintosh.html**
Intro. to WWW	**http://www.eit.com/web/www.guide/**
Web66	**http://web66.coled.umn.edu/**

New Technologies

Apple	ftp://ftp.info.apple.com
DTS	http://www.info.apple.com/dev/dts.html
see also	http://www.austin.apple.com
Dylan	http://www.cambridge.apple.com/
see also	ftp://cambridge.apple.com/pub/dylan
see also	http://legend.gwydion.cs.cmu.edu:8001/dylan
see also	news://comp.lang.dylan
Kaleida	http://www.kaleida.com/
OpenDoc/Bento/SOM	**http://www.cilabs.org/pub/cilabs/tech/**
OpenTransport/TCP	**gopher://seeding.apple.com**
Taligent	http://www.taligent.com/

Other Programmer Resources

Ada	ftp://ftp.seas.gwu.edu/pub/ada
Applescript	ftp://gaea.kgs.ukans.edu/applescript
Get1Resource	http://www.asel.udel.edu/~haynes/g1r.html
Robert Lentz	http://www.astro.nwu.edu/lentz/mac/programming/home-prog.html
Lisp, MCL	**http://www.digitool.com/**
General	http://www.cs.rochester.edu/u/miller/alu.html
MacHack	**http://www.consensus.com:80/~machack/**
see also	**http://www.macgroup.com/MacHack.html**
MacTech	**ftp://ftp.netcom.com/pub/xp/xplain**
Matthias Neeracher	http://err.ethz.ch/members/neeri.html
nick.c	*good for beginning Macintosh programmers!* http://www.pitt.edu/~nick/
François Pottier	http://acacia.ens.fr:8080/home/pottier/index.html
also	news://comp.sys.mac.digest
Digests archive	**ftp://ftp.dartmouth.edu/pub/csmp-digest**
Jon Pugh (AppleScript)	ftp://ftp.netcom.com/pub/jo/jonpugh/homepage.html
Paul Robichaux	http://www.iquest.com/~fairgate
Source code	http://www.info.apple.com/dev/devinfo/macsourcecode.html
also UMich	ftp://mac.archive.umich.edu/mac/development/source/
also M. Neeracher	ftp://ftp.switch.ch/software/mac/src/HTML/Welcome.html
TCL stuff	ftp://daemon.ncsa.uiuc.edu/TCL

Vendors, Products and Miscellaneous

Alpha (text editor)	**http://www.cs.umd.edu/~keleher/alpha.html**
BBEdit	ftp://ftp.netcom.com/pub/bb/bbsw
Celestin	ftp://ftp.teleport.com/vendors/cci/apprentice
CIL	http://www.cilabs.org/
Digitool	**http://www.digitool.com/**
Dilbert	**http://gnn.interpath.net/gnn/arcade/comix/graphics/Dilbert.gif**
EduPage newsletter	**mailto:listproc@educom.edu** in the body of the message put: subscribe edupage *your name*
Just Some Guy	**http://www.spies.com/greg/**
Macintosh Vendor Directory	http://rever.nmsu.edu/~elharo/faq/vendor.html
Mac Netswitch	http://www.nd.edu/~dwalton1/
MacNosy	ftp://ftp.netcom.com/pub/ma/macnosy
Metrowerks	http://www.iquest.com/~fairgate/cw/cw.html
NeoLogic	**http://www.neologic.com/~neologic/**
QKS/Smalltalk	http://www.qks.com
QUED/M	**ftp://ftp.nisus-soft.com/pub/nisus**
see also	**http://www.nisus-soft.com/~nisus**
QuickCam	**http://www.jungle.com/msattler/sci-tech/comp/hardware/quickcam.html**
Symantec	**ftp://devtools.symantec.com/macintosh/updaters/devtools**
TidBITS newsletter	http://www.dartmouth.edu/pages/TidBITS/TidBITS.html
see also	news://comp.sys.mac.digest
to subscribe	**mailto:info@tidbits.com**
Time Tracker	ftp://ftp.maui.com/pub/mauisw
UserLand AutoWeb	**http://www.hotwired.com/Staff/userland/**

URLs in boldface are new to this column, or have changed recently. Names in boldface are sites notable for their quality or timeliness. mailto: URLs work with browsers like Netscape. To use them manually, send mail to the part after the colon.

Be sure to keep up on the Portable Network Graphics standard if you work with graphics online. GIF's days are numbered: **http://sunsite.unc.edu/~boutell/png.html**

Planning to host a Web site? Be sure to read our MacHTTP article this month, and check out these important sites:

http://www.uwtc.washington.edu/Computing/WWW/Macintosh.html
http://web66.coled.umn.edu/

If you happen upon a worthy Internet resource, please send it to us at `online@xplain.com`.

COMPILER SETTINGS

By Bill Karsh, billKarsh@aol.com

Universal Headers 2.0a3 *f*

The Devil's In The Details

Don't get surprised by the Universal Headers

There is a particularly insidious problem in Apple's Universal Headers. Whether you are affected at all, and how devastating it is, depends upon how you use their headers in your projects. Fortunately, the problem only affects 68K code generation. Here are the details, and what you can do as a work-around until new headers become available.

Which Header Versions?

The problem appears in the headers I got with Think C/C++ version 7, although these headers themselves do not seem to have a version number. The problem persists through the latest headers I got with MetroWerks CodeWarrior 5. These are version 2.0a3. Undoubtedly, you are using an affected version (I think they all are).

Structure Alignment Problem

The problem is a structure alignment conflict. Most compilers today let you apply some options regarding how structures are defined, throughout the whole of a project, or while certain pragmas are in effect. For example, you find a pop-up menu in CodeWarrior's Processor Preferences panel, that let's you select options called "68K," "68K 4-byte," or "PowerPC." You can also use pragma statements like these:

```
#pragma options align=mac68k
#pragma options align=mac68k4byte
#pragma options align=native
#pragma options align=power
#pragma options align=reset
```

These can be used to override a globally set option. I believe that the last option stays in effect until either the next alignment statement, or the end-of-file is encountered.

In Symantec C++, the Compiler Settings page has a radio button group called "Struct Field Alignment," that sets alignment options for a project.

Generally, the chosen alignment determines how much padding (extra space) is inserted into structures between fields, and how much padding is appended to the end of structures. Deciding which option you would want is a matter of trading-off between compactness of the data (less space) and higher performance (fields aligned on natural addresses for the field's data-type). We won't go into those decisions here. What is immediately essential is understanding two things.

First, these options are available to you (settable in an options page or by pragma statements) independently of whether you are generating PowerPC (ppc) code or 68K code.

Second, structures defined by the Mac OS, that is, defined in Apple's headers, must have no extra padding in them. When the compiler comes across the definition of any Mac OS structure, like GrafPort, BitMap, or whatever, the current alignment option must be mac68k. If the current setting were anything else, your code, and the system software, would disagree on the offsets to various fields in a GrafPort, for example. That's a disaster.

Inside Mac Headers

If you look inside virtually any Apple header, you will see one of the following examples near the top of the file:

Continued on page 83

Bill Karsh – You may remember Bill from his two-part series last year on PowerPC assembly language and from his participation in the Programmer's Challenge. He's kept his eyes open, and that's good for all of us who use the new Apple interface files.

MacRegistry™

Developer Job Opportunities

If you are a Macintosh developer, you should register with us! We have a database that enables us to let you know about job opportunities. When we are asked to do a search by a client company the database is the first place we go. There is no charge for registering. The database service is free. Geographic Coverage is nationwide.

Marketability Assessment - To get a specific feel for your marketability send a resumé via Email or call. You may also request a Resume Workbook & Career Planner.

Discreet - We are very careful to protect the confidentiality of a currently employed developer.

Scientific Placement is managed by graduate engineers, we enjoy a reputation for competent & professional job placement services and we are Mac fanatics.

800-231-5920 | das@spi.com | Fax 713-496-0373

Scientific Placement, Inc.

MT, Box 19949, Houston, TX 77224 713-496-6100 Fax: 713-496-0373
MT, Box 71, San Ramon, CA 94583 510-733-6168 Fax: 510-733-6057
MT, Kenmore Station, Box 15225, Boston, MA 02215 617-424-8372 Fax: 617-424-7158
AppleLink: D1580; Compuserve: 71250,3001, eWorld: spi, AOL: davesmall

The Trattner Network

The Trattner Network, a digital talent source, is looking for experienced Macintosh developers for a variety of consulting opportunities. Current projects include development in the areas of OpenDoc, Newton, Power PC, Metrowerks, PowerBook, MacApp, 4D and many others.

We have *urgent* needs for:

- Software Developers
- Hardware/Firmware Engineers
- QA/QC Professionals
- MultiMedia Developers
- Project Coordinator/Manager
- Network Professionals

The Trattner Network brings 10 years experience in the Macintosh consulting and placement industry, offering a unique blend of humanistic interaction and technical knowledge. Positions in Northern California and Nationwide.

If you are looking for the chance to enhance your skills, team with the best, and make lots of money, send, fax or link your resume to:

TTN
Attn: Nyla Miller
170 State St., Suite 240, Los Altos, CA 94022
Phone (415) 949-9555 • Fax (415) 949-1026
AppleLink: trat.net • email: nmiller@tratnet.com

MAC PROFESSIONALS

Manpower Technical, a leader in the Technical Services industry, has current and upcoming contract openings for experienced MAC personnel with the following expertise: Software Development, Network Administrators, Desk Top Publishing, Help Desk, Applications Support, Technical Writers/Editors.

Positions are in California and throughout North America.

Interested persons are encouraged to send their resume to:

Manpower Technical
ATTN: Dept. MT
P.O. Box 2053
Milwaukee, WI 53201

(800) 558-6992
Fax # (414) 332-0378

MAC ENGINEERS

Application Resources, Inc. is one of the primary technical contracting companies providing services to Apple Computer. We have immediate long-term contract opportunities for Macintosh engineers in a variety of disciplines, including product development, quality assurance and telecommunications, utilizing MacApp, TCL, Virtual User, C/C++, MPW, NewtonScript, Metrowerks and PowerPC.

Contact: Nancy Falkenburg
Bob Gates

Application Resources
Mountain View, CA
Tele. 408-245-9899
Fax 800-433-6121
Internet: resumes@appres.com

MAC PROGRAMMERS

Looking for
that perfect contract?

We specialize in the Macintosh market and have the experience and ability to find you that opportunity you've been waiting for. We have openings for Mac Programmers with MacApp, C++ and Toolbox exp. NewtonScript Programmers are also needed.

West Valley Engineering
1183 Bordeaux Drive
Sunnyvale, CA 94089
(408) 734-4338 fax
Applelink: WVE.SFTWRENG
Internet:
westvalley@cup.portal.com

The Devil's in the Details continued from page 81

```
#if defined(powerc) || defined (__powerc)
#pragma options align=mac68k
#endif
```

or

```
#if GENERATINGPOWERPC
#pragma options align=mac68k
#endif
```

Near the bottom you will see one of:

```
#if defined(powerc) || defined (__powerc)
#pragma options align=reset
#endif
```

or

```
#if GENERATINGPOWERPC
#pragma options align=reset
#endif
```

The intent here is to force the alignment type to mac68k while any Mac OS structures are defined. Then, align=reset puts the alignment back to whatever was in effect before. Can you see the problem? The pragma statements are made conditional upon what type of code is being generated, ppc or 68K. If you are generating ppc code only, there is no problem. If you are generating 68K code, the pragmas are not executed. You've got a bad problem in 68K code if, when the headers are compiled, the current alignment option is not mac68k. The pragma statements in the Apple headers were not supposed to be conditional upon anything. I'm guessing here, but it looks like a clear case of macro-mania-a-go-go.

Why Didn't I See That?

I know, right now you are saying "how can any of my code work at all if this is true? Why didn't I catch this immediately!?" There are several sneaky ways this problem might have escaped your notice thus far. For one thing, 68K compilers typically ship with mac68k as the default alignment option. If you never changed that, you would have been spared. Another possibility involves precompiled headers.

Your environment was shipped with precompiled headers. That is, many, but not all, of the Apple headers were precompiled at the factory into a giant header called MacHeaders68K (CW) or MacHeaders (Think), or something similar. Such a file is usually included in a global preferences dialog. That would be the Language panel in CW, or the Prefix page in Think's dialog. There may be separate precompiled headers for C and C++. Again, chances are that you never removed it. Back at the factory, they compiled these precompiled files using the mac68k setting. Therefore, all the structures defined in there are fine. However, not all of the Apple headers are typically included in these things, only the most commonly used. This keeps symbol table sizes down, and speeds compile times.

Any time you find a need to explicitly include an Apple header in your sources, and it's not among those precompiled at the factory, you are letting yourself in for it. You have to do something to make sure that when the compiler reads that included header, the alignment is mac68k. You have to take these steps yourself because the provision to do this automatically in the headers themselves is incorrectly coded.

What Do I Do?

If you understand what's happening, you can choose whatever method you like to adjust for this; whatever works and is convenient. Let's suppose you need Timer.h, and it's not currently precompiled. Here are some possibilities:

Re-precompile MacHeaders68K, or whichever variant your project needs. Uncomment Timer.h in the source file that generates the precompiled header. Make sure the project's global option is "68K" (CW) or "Align to 1 byte Boundary" (Think). Precompile a new MacHeaders, and include it in your working project's preferences dialog.

Include Timer.h in a source file like this:

```
#pragma options align=mac68k
#include <Timer.h>
#pragma options align=reset
```

Repair the Apple header directly. Change this:

```
#if GENERATINGPOWERPC
#pragma options align=mac68k
#endif
```

to this:

```
#pragma options align=mac68k
```

Make the alignment options unconditional, as they were supposed to be. Do this for the reset lines too!

I know this will save you days of debugging and thousands of dollars of development costs. Could you send me just one dollar as a thank you? Worth a try.

DIALOG BOX

By Scott T Boyd, Editor

Concerned By The Rising Tide

I have developed several large programs for use in my engineering classes during the past six years. Although I am not a professional programmer, I believe my programs to be of commercial quality and at least one has been fairly widely distributed among universities. My programs have all been developed in Pascal with separate versions to operate within the Macintosh, DOS and Windows environments. I use Borland's Pascal on the PC and Think Pascal on the Macintosh. I use objects extensively for both operating system needs and for my own applications.

I have been watching with some concern as the popularity of Pascal has declined in favor of C/C++. I decided to learn C/C++ so that I could see for myself why it is gaining in popularity. The basic concepts and constructs of Pascal and C++ are quite similar and, after getting used to the syntax, it was not difficult to translate code from Pascal to C/C++.

Having now translated several small programs and one large application, I thought that you might be interested in a summary of my assessment of the two languages.

One feature provided by Pascal that I sorely miss in C/C++ is nested procedures. In Pascal, a function or procedure can be placed within another function or procedure so that it visible only to the external routine. Variables declared within the nested routine are local to that routine and not accessible to the external routine whereas variables declared in the external routine are accessible to the nested routine. I find this nesting capability to be very helpful and I often nest my routines four or five levels deep. I find the nesting parallels my thinking and aid programming. As with objects, nested routines tends to encapsulate code and data and make the overall function of a routine more clear. Nested routines are not allowed in C++. I was forced to de-nest all of my routines in translating from Pascal to C++ resulting in a large number of routines all at the same level. The difference in variable scoping complicated this process.

A serious disadvantage of C++ that I encountered is that the time required to compile and link was much greater than in Pascal. Because I program in a manner in which I make a change and then test it, I found the speed difference to be irritating. The difference exists on both the Macintosh and the PC even though the Pascal and C++ compilers were from the same vendors. I understand that pre-compiled headers can be used to speed the compile/link process and I experimented with them, but I never approached the compile and link speed of Pascal. Why is it that Pascal does not require pre-compiled headers? I truly don't understand why this difference exists. Does it reflect poor compiler design or an innate problem with C++?

I also noted that small programs compiled as code resources were much larger when compiled in C++ than in Pascal. This size difference is probably due to C++ including some libraries but I was unsuccessful in finding a way to reduce the size. In any case, I did not observe significant execution speed differences between programs compiled in Pascal and C++.

In summary, I have not found any capabilities in C++ that I need and do not already have in Pascal. I find the Pascal environment preferable, and I found the auto-formatting and integrated debugging provided by Think Pascal on the Macintosh to be much more friendly than the C/C++ environment. I'm curious to know if I am missing something or whether others have had similar experiences.

– S.A. Klein, University of Wisconsin Madison
klein@engr.wisc.edu

OpenDoc Draws Some Fire

I read the OpenDoc and SOM articles in the January MacTech. I have some comments, mostly skeptical ones.

It seems to me, OpenDoc was born as a succession of ideas:

- Bundle MacApp into the MacOS ROM and System 7.5, and
- change this class library into robust objects for everyone.
- Emphasize document mobility among apps using these objects.
- Call it "Documents On-top-of Classes", or DOC.

Now, the Apple budget guys come into the picture.

- No more free lunch, so it doesn't go into the ROMs.
- No more funding either, so spin it off to a 3rd party: CIL.ORG.
- IBM wants a kickback for PowerPC, so push SOM.

The Apple Evangelists have a field day!

- Model all users as "document fiddlers": readers, writers, editors.
- Model all data processing as "document processing".
- Model all software vendors (eg, Microsoft) as "editor vendors".
- Hook up with other popular hype floating around:
  – Cobra Objects Really Biting Applications (CORBA)
  – Subtle Object Mangler (SOM)
- Rename the project OpenDoc, reflecting the open question of "Who pays for all of this?"

Now we are faced with some questions. What is Apple's strategic perspective on this; e.g., will they push OpenDoc as a replacement for MacApp?

Will OpenDoc be an optional, extra-cost item (like System 7 Pro), or a developer giveaway on the ETO disk? Or perhaps just a standard for other developers since Apple doesn't manufacture any "editors" except TeachText and ResEdit?

What is Microsoft's perspective on all this, given that they are probably the biggest producer of "commercial editors" (formerly known as "business software")? The MacTech article says the OLE is just a subset of OLE – how is this enforced, and does Microsoft agree?

Before Apple pushes the hype of OpenDoc, shouldn't it first come up with some systematic scheme about parts? What are standard "parts" anyway? If each developer whips up their own set of "parts", how does this lead to interoperability?

Is Apple committed to supporting this technology in the future, or just committed to advertising and evangelizing the idea?

As an analogy, compare "handlers and parts" with the "Communications Toolbox and Tools". Apple did OK with the first one, but skimped on the second, and the result is a less useful system. Certainly modem communication has not become easier as a result of the CommToolbox. What prevents this from happening with OpenDoc?

More brickbats – Compare the OpenDoc plan with the MIT X-Windows project, which also started out as a simple idea (a windowing system for Unix) but ended up so complex it rivals Vax/VMS. Problems include being the lowest common denominator to all types of hardware, too many toolkits but no standard of functionality within them, and implementation by student/researchers with little vested interest in simplicity. The result is a system which works well, but don't try to program it directly without lots of available staff time. What will keep OpenDoc from becoming so complicated when it's finished?

Compare the OpenDoc plan with Ada, the programming language of the future just ten years ago. Ada has all sorts of "necessary" features for dealing with interrupts, multi-tasking, events, and real-time structures. Too bad most operating systems can't or don't or won't provide all the necessary support for these things, otherwise we'd be

Continued on page 86

NEWSBITS

By Scott T Boyd, Editor

What's That Gestalt Value?

Rene Ros (rgaros@bio.vu.nl) released version 2.8 of the Gestalt Selectors List (GSL). It lists all sorts of information about the Gestalt Manager, but mainly about selectors and the meaning of the returned values. The Gestalt Manager is part of the Apple Macintosh System Software to enable programmers to determine the availability of certain software and hardware. You can obtain the latest version in several ways.

email: gestalt-selectors-list-request@bio.vu.nl
with subject: archive get recent/gestalt-selectors.etx
or to get the compressed version: archive get recent/gestalt-selectors.sit.hqx
ftp: ftp://sumex-aim.stanford.edu/info-mac/dev/info/gestalt-selectors-xx.hqx

You can also use any of its mirror sites.

World Wide Web: http://www.astro.nwu.edu/lentz/mac/faqs/source/gestalt.html
CompuServe: (GO MACDEV) in the Forum Business/Help (1) section.
America OnLine: MDV/Documents and Proposals directory.

Contributions (new info, remarks, etc.) for the list can be send to gestalt-selectors-list@bio.vu.nl

Bare Bones Does GX On The Internet?

Bare Bones Software, Inc. announced the immediate availability of version 3.1.1 of BBEdit, their popular and critically acclaimed text editor. This new version incorporates several significant enhancements to its feature set and capabilities. The update also features assorted fixes and performance improvements.

BBEdit now supports Quickdraw GX printing services. This enables BBEdit users to use the advanced print-job-control capabilities of Quickdraw GX, as well as use third-party extensions which enhance printing capabilities, such as the Peirce Print Tools.

BBEdit 3.1.1 features the ability to interact with Internet "helper" tools such as World-Wide Web browsers, FTP clients, and Usenet news readers. A new menu provides direct access to helper applications, and the new version provides the ability to resolve URLs simply by clicking on them or choosing a menu command. A "View HTML File" command assists in the previewing of HTML documents with the user's chosen Web browser. The CD-ROM includes new HTML markup tools.

BBEdit 3.1.1 is available immediately as a free update for users of BBEdit 3.1. The updater can be found in a variety of Mac related archive sites on the Internet, in the file libraries of AOL, eWorld, and CompuServe, and on Bare Bones Software's own FTP site (ftp://ftp.netcom.com/pub/bb/bbsw/bbedit-311-upd.hqx). Finally, Bare Bones Software will be sending out updater disks to all customers who purchased BBEdit 3.1 directly from them. Users of older commercial versions of BBEdit can upgrade for US$39 plus shipping. Questions are welcome at bbsw@netcom.com or (508) 651-3561.

It's Magic! It's CodeWarrior! It's Both.

Metrowerks (VSE,MSE:MWK) and General Magic (Nasdaq:GMGC) announced their agreement for Metrowerks to introduce CodeWarrior Magic, a Macintosh-hosted toolkit for Magic Cap applications development. CodeWarrior Magic will be the first publicly-available toolkit for the Magic Cap platform, and its availability will mark the opening of the Magic Cap developer platform to developers outside of General Magic's current developer program. Programmers proficient in C who purchase CodeWarrior Magic will be able to create applications for Magic Cap personal communicators.

The first version of CodeWarrior Magic, developer release DR1, will be supported on both 68K and Power Macintosh. It will be based on Metrowerks' CodeWarrior MPW tools and on General Magic's object-oriented development environment, the environment used by present Magic Cap developers to produce Magic Cap applications. CodeWarrior Magic is intended to allow developers to develop and debug Magic Cap applications directly on a Macintosh, giving them the opportunity to do much of their engineering work on a Macintosh prior to downloading applications to a Magic Cap personal communicator for final debugging. A subsequent version of CodeWarrior Magic is intended to contain General Magic components integrated into Metrowerks' new object-oriented development environment.

CodeWarrior Magic DR1 will be priced at $299, and will include a free upgrade to the next release of CodeWarrior Magic (DR2) and two subsequent releases, as well as full network support from Metrowerks' Technical Support group. Metrowerks plans to introduce CodeWarrior Magic DR1 right around the time you see this.

Metrowerks info: sales@metrowerks.com.
Magic Cap developer info: dev-info@genmagic.com.

SourceSafe Source Code Management

Microsoft announced the availability of Microsoft SourceSafe 3.1, its version control/configuration management system. The project-oriented approach to source-code control promotes code reuse and management of multiple releases while preventing accidental code loss or alteration.

It allows backtracking to previous versions, branching and merging, and offers security and audit-trail capabilities. It can be used as a central development manager by software developers using other Microsoft tools. Microsoft acquired SourceSafe as part of its purchase of One Tree Software, Inc. in November.

Microsoft SourceSafe supports MS-DOS, Windows, Windows NT and Macintosh. It's available now from software resellers for approximately $499 for a single-user license.

Dialog Box continued from page 84

using it today. The OpenDoc plan appears to require standardized features across operating systems (eg: network clipboards) which just aren't present today, at least, not in an "open" sense. Ask yourself again, what is Apple's role in all of this, and what guarantees us that Apple won't change its commitment like it did with Bedrock.

Compare the OpenDoc plan with OSI (Open Systems Interconnect), the new reference standard network protocol, which has made itself obsolete in just a few years. Why should we believe that OpenDoc will be any more of a universal standard than OSI ?

Compare the OpenDoc plan with Apple's OpenTransport. Unlike OpenDoc, the OpenTransport plan has specific goals, a specific plan of implementation and timeframe, and a specific support commitment from Apple. The marketing side of OpenTransport exactly matches its technical specifications. On the other hand, the OpenDoc literature talks about "Apple's approach towards reducing the complexity of computing today", and "providing users with a new level of computer power, flexibility, and ease of use". Sounds like they're talking about the Mac itself, until you read the fine print about compound documents.

The OpenDoc blurb talks about user interfaces and standard user operations, yet from what I can see, these implicit issues lack any sort of formal model or standard definition. Just what does standard text editing mean anyway, and why should I switch away from MS-Word to some new editor?

So far, OpenDoc is a philosophy rather than a product. Example: consider a C++ program as a document. There are a variety of handlers available: editors: MPW, vi, emacs, Think Project, BBEdit; Lint'ers; parenthesis matchers; compilers for the code. Yet the "document" – source code for some program – is already "sharable" with the first set, and usually incompatible with the last set. That is, a large body of code typically cannot be sent to just any other compiler, certainly not cross-platform, without manual intervention, setting compilation flags, adjust the code a bit, etc. What value does OpenDoc add to this scenario?

I suppose I have the wrong idea – I'm not supposed to think of "source code" as a document. Instead, the main focus of my work should be writing letters with text processors, and this entitles me to subscribe to the OpenDoc hype. Unfortunately this isn't true, and therefore I'm concerned that OpenDoc will just lead to a more complicated development environment with little added benefit.

Well, these are my observations. – *John Buehrer, jdb@ecofin.ch*

Apple's Jens Alfke responds:

John Buehrer seems to have gotten a very mistaken impression of what OpenDoc is and what it aims to do; this is understandable, since it's hard to describe a complex system in a short article.

The three major assumptions he makes are that OpenDoc is supposed to be a universal solution for all types of software; that it is a framework (like MacApp or PowerPlant); and that it is vaporware without an implementation. None of these is true.

- OpenDoc does *not* claim to "model all data processing as 'document processing'" and we do *not* expect every type of software to become a part editor. However, most of what users do with their computers is document processing – text, spreadsheets, drawings, page layouts, et cetera – so focusing OpenDoc on documents doesn't seem like much of a restriction. But there is certainly still room for traditional apps and other types of software in the world.
- OpenDoc is not a framework in the traditional sense. The developer does not assemble classes provided by OpenDoc to produce a piece of software; rather, OpenDoc assembles the developer's editors at runtime to produce a document. In other words, OpenDoc lives in the spaces between editors, not within them. It's more like that Macintosh Toolbox in that regard.

Frameworks can certainly take advantage of OpenDoc (and MacApp 3.5 and the OpenDoc Parts Framework will do so, as may PowerPlant) but the tasks they perform are orthogonal. OpenDoc itself does the minimum it can do to guarantee smooth operation of multiple components in a document; everything beyond that is a job for the developer or for a framework.

(The history of OpenDoc really had nothing to do with the MacApp project; instead, it grew out of an investigation of how to extend technologies like Apple Events, the Edition Manager and Bento into a true compound document architecture.)

- Mr. Buehrer also seems concerned whether OpenDoc really exists and whether Apple is truly committed to it. OpenDoc has an large engineering effort, of which I'm a part, and a public delivery schedule. We've done a lot more than just "advertising and evangelizing the idea"; we've been working on the implementation since 1992. The software on the CD should attest to this. Moreover, Apple executives from Spindler on down are committed to OpenDoc; they rightly see it as an absolute necessity if Apple is to maintain its technological leadership and its control of its platform. From my position in the engineering trenches I see every indication that OpenDoc is one of the most high-priority projects at Apple.

He then asks what will keep OpenDoc from becoming unusably complex. Some complexity is inevitable, but what developers who've used it tell us is that the architecture makes sense and that it's much cleaner than OLE. Take a look for yourself; the APIs on the CD are pretty close to final (in the next release they truly will be frozen.)

He sums it up by saying that "So far, OpenDoc is a philosophy rather than a product." One might get this impression by reading just the (admittedly rather marketing-oriented) article in the magazine. But if you go beyond that and examine the more technical documentation provided on the CD or on the Internet (by FTP from cilabs.org) we hope you'll see that OpenDoc is solid, implemented, and well on its way to completion. – *Jens Alfke, OpenDoc Engineering Team*

An Interesting Design Philosophy

Here's a Word 6 "glitch" I thought you might find interesting. Go into Word and try to type control-Q or control-T (in Chicago font, the command key symbol and the apple symbol, respectively). Of course, nothing happens. Someone at Microsoft decided that the control keys shouldn't be typeable! So I got onto the MSWord forum and asked for help. These folks are great. To their credit, every question I've posed them has been answered promptly and accurately, if not necessarily to my satisfaction. However, below is their response to my query. As Dave Barry would say, "I'm not making this up!"

– *Dave Mark*

from the MSWord forum:

Word 6 now inserts special characters with the Symbol dialog. To access the dialog, choose Symbol from the Insert menu and click on the Symbols tab. The drawback to using the Symbol pallet is that special characters below the value of 32 (check Appendix A to determine what value is associated with a character) in the Macintosh character set are not available. Unfortunately, for characters 0 through 31, you'll need to go through a couple extra steps to insert them with a keystroke.

To get around this problem, insert the Command character and the Apple characters in Word 6.0 using fields, and then for convenience, make the character into a glossary entry, and assign your own keystroke to them.

To use the symbol field, Choose Field from the Insert menu, click on "Equations and Formulas" under Categories, then click on "Symbol" under Field Names. Place the cursor in the text box next to the word "SYMBOL", type in the value for the character and a space (see Appendix A: you'll see the Command character has a value of 17), then click on the Options button and add the \f switch. Then click in the text box after the \f switch and type in the name of the font you want, in your case, "Chicago" without quotes. Click OK. Then click OK again. You should see the Command character at this point. If you do not see it, place the cursor on the field (It might look like {SYMBOL 17 \f Chicago *MERGEFORMAT}) and then press SHIFT F9. This keystroke changes the view of the field to the Command character.

To make the process more convenient for future use, select the symbol, choose AutoText from the Edit menu, type a name in the Name box; for example, "command", and click the Add button. Then, to assign the keystroke, choose Customize from the Tools menu, on the Categories side select AutoText (it's near the bottom of the list), select the AutoText entry on the right side, place the cursor in the "Press New Shortcut Key" box, press the keystroke CONTROL Q and click the Assign button. The next time you need to use the character, just press CONTROL Q.

Tips & Tidbits continued from page 96

you put the Apple Menu Items alias onto the desktop, System 7.5 maintains automated folders, so the Apple Menu Items Alias can be used to get to "Recent Documents". The "Recent Servers" can be used to log onto to any server without leaving the application that you are running.

– Mark Tillinghast
XXCAL, Testing And Systems Division
mark_tillinghast@lamg.com

Debugging For Fun and Profit

When you have routines that return OSErrs, finish them with the following bit of code:

```
...
ReportError ( err );
return err;
}
```

which is defined as:

```
#ifdef qDEBUG
#define ReportError(err) do { if (err != noErr ) DebugStr ( "
\pError" ); } while (false)
#else
#define ReportError(err) /* nothing */
#endif
```

With this macro, you can tell every time that one of your routines returns an error, and it's easy to follow your error handling code.

– Marshall Clow
Aladdin Systems

What's My Modifier?

To quickly get the status of modifier keys, use these simple lines:

```
typedef struct kbd_bits {
  int command : 1;
  int : 5;
  int space : 1;
  int tab : 1;
  int : 4;
  control : 1;
  option : 1;
  capslock : 1;
  shift : 1;
} kbd_bits;

#define KeyModifiers (*(kbd_bits *)0x17A)
```

Then you can say simply:

```
if (KeyModifiers.control) {
  Debugger(); // drop into a debugger
}
```

This works across all Macs, including foreign versions and keyboard maps. For maximum compatibility with a minimal speed hit, call GetOSEvent with a null event mask and examine the modifiers field of the event returned.

– Jorg Brown
The Mac Group

MACTECH MAGAZINE PRODUCTS & ORDER INFORMATION

E-mail, Fax, write, or call us. You may use your VISA, MasterCard or American Express; or you may send check or money order (in US funds only):
MacTech Magazine, P.O. Box 250055, Los Angeles, CA 90025-9555.
Voice: 310/575-4343 • Fax: 310/575-0925

If you are an e-mail user, you can place orders or contact customer service at:
- **AppleLink:** MT.CUSTSVC
- **CompuServe:** 71333,1063
- **Internet:** custservice@xplain.com
- **America Online:** MT CUSTSVC
- **GEnie:** MACTECHMAG

SUBSCRIPTIONS

US magazine with domestic source code disk: $124 for 12 issues
Canadian magazine with Canadian source code disk: $136 for 12 issues
International magazine with international source code disk: $194 for 12 issues

CD-ROM

MacTech CD-ROM, Volumes I-IX: Includes over 1100 articles from all 103 issues (1984-1993) of MacTech Magazine (formerly MacTutor). All article text and source code. Now in THINK Reference format. The CD includes Symantec's THINK™ Reference 2.0, working applications with full documentation, product demos for developers and more. See advertisement, this issue: $199. Upgrades $69, e-mail, call or write for info.

BOOKS

The Best of MacTutor, Volume 1: **Sold Out**
The Complete MacTutor, Volume 2: **Sold Out**
The Essential MacTutor, Volume 3: $19.95
The Definitive MacTutor, Volume 4: $24.95
The Best of MacTutor, Volume 5: $34.95
Best of MacTutor Collection, Volumes 3 – 5: $69
Best of MacTutor, Volumes 6, 7, 8 & 9: Not available

DISKS

Source Code Disks: $8 each
Topical Index (1984-1991) on disk: $5

MAGAZINE BACK ISSUES

Volumes 3, 4, 5, 6, 7, 8, 9 and 10: $5 each (subject to availability)

California residents include 8.25% sales tax on all software, disks and books.

Allow up to 2 weeks for standard domestic orders, more time for international orders.

PLEASE NOTE

Source code disks and journals from MacTech Magazine are licensed to the purchaser for private use only and are not to be copied for commercial gain. However, the code contained therein may be included, if properly acknowledged, in commercial products at no additional charge. All prices are subject to change without notice.

10% OFF ALL BOOKS!

3RD PARTY PRODUCTS

MACTECH EXCLUSIVES

MacTech Magazine is your exclusive source for these specific products:

Ad Lib 2.0 The premier MacApp 3.0 compatible ViewEdit replacement. A powerful user-interface editing tool to build views for MacApp 3.0 and 3.1. Ad Lib allows subclassing of all of MacApp's view classes including adorners, behaviors, and drawing environments. String and text style resources are managed automatically. Alternate display methods, such as a view hierarchy window, allow easy examination of complex view structures. Ad Lib includes source code for MacApp extensions that are supported by the editor – buttons can be activated by keystrokes, behaviors can be attached to the application object, and general purpose behaviors can be configured to perform a number of useful functions. Run mode allows the user to try out the views as they will work in an application. Templates can be created to add additional data fields to view classes. Editing palettes provide fast and easy editing of common objects and attributes. Works with ACI's Object Master (version 2.0 and later) to navigate a project's user interface source code. $195

NEW! **FrameWorks Magazine:** $8/issue, subject to availability.

NEW! **FrameWorks Source Code Disk:** $10/issue, subject to availability.

NEW! **Five Years of Objects CD-ROM:** FrameWorks archives and source code from April 1991 to January 1993, plus selected object-oriented publicly available software and demos. $95

MADACON '93 CD-ROM: The highlights of MADACON '93, including Mike Potel on Pink, Bedrock, MacApp, OODLs, and more. Slides, articles, demos, audio, and QuickTime. $95

NEW! **MAScript 1.2** adds support for AppleScript to your MacApp 3.0.1 and 3.1 based applications. Make your application scriptable and recordable by building on a tried and tested framework for object model support. MAScript dispatches Apple events to the appropriate objects, creates object specifiers, and makes framework objects like windows and documents scriptable and recordable. Sample application shows you how to begin adding support for scripting and recording. MAScript includes complete source code. Install MAScript by modifying one MacApp source file, then adding another to your project. Future versions of MacApp will incorporate MAScript, so MAScript support you add now will work in the future. $199

The Mjølner BETA System is a software development environment supporting object-oriented programming in the BETA programming language. BETA is uniquely expressive and orthogonal. BETA unifies just about every abstraction mechanism – including class, procedure, function, coroutine, process and exception – into the ultimate abstraction mechanism: the pattern. BETA includes: general block structure, strong typing, whole/part objects. The compiler: binary code generation, automatic garbage collection, separate compilation, interface to C, Pascal, and assembler. The system: persistent objects, basic libraries with containers classes, platform-independent GUI application frameworks on Unix, Mac and Windows NT, metaprogramming system. The tools available on Unix: the hyper structure editor supporting syntax directed editing, browsing, etc., and the source code debugger are currently being ported to the Macintosh system. The Mjølner BETA System for Macintosh requires MPW (basic set) 3.2 or later. Package containing compiler, basic libraries, persistent store, GUI framework, and comprehensive documentation. (Other packages are also available) $295

NEW Version! **Savvy 1.1** OSA support includes attachability, recordability, scriptability, coercion, in addition to script execution, idling and i/o. Apple event support includes complex object specifiers, synchronous/asynchronous Apple event handling, and Apple event transactions for clients and servers. The Core Suite of Apple event objects is supported including the application, documents, windows, and files. Documentation includes technology overview, cookbook, and sample code. $250 Savvy now supports MPW 3.1, 3.11 and continues to support 3.01, as well as supporting Metrowerks Code Warrior. **This month only, special offer –** All Savvy versions include free copy of Savvy QuickTime!

NEW! **More Savvy** includes all Savvy features plus Apple event support for all sub-classes of TEventHandler with extensive view support. Apple event support for text includes text attributes and sub-range specification. Recordability supports additional actions, and coercion includes additional types. Additional client and server Apple events. $450

NEW! **Super Savvy** includes all More Savvy features plus compile, edit, and record scripts using built in script editor. View template editors, like Ad Lib, can attach scripts to view objects and modified scripts are saved with the document. Script action behavior allow quick access for executing and editing scripts attached to views. Text to object specifier coercion plus more. $700

NEW! **Savvy QuickTime** Requires Savvy, More Savvy, or Super Savvy. Includes QuickTime, Apple event and view template support. Movies come out of the box ready to play, edit, and react to Apple events. They can be included in any view structure, including templates, and are displayed in the scrap view. Movie controls include volume, play rate, looping mode, display style, and other characteristics. $250

NEW! **Savvy DataBase** Requires Savvy, More Savvy, or Super Savvy. $250

MAIL ORDER STORE

MacTech Magazine is your exclusive source for available back issues of SFA's magazine, source code disks and assorted CD's. Call for more info and pricing.

BOOKS

NEW! America Online For Dummies™ by John Kaufeld. "Driver's Education" for this wildly popular on-line service – covering everything from the main menu to the mail groups for Windows, DOS, and Mac platforms. Includes a coupon for free usage time on AOL for first-time users. ~~$19.99~~ **$17.95**

NEW! C For Dummies,™ Volume 1 by Dan Goodwin. Finally! A hands-on, step-by-step tutorial for learning the essentials of programming in C. ~~$19.95~~ **$17.95**

NEW! C++ For Dummies™ by Stephen R. Davis is the all-in-one reference that gets you comfortable with object-oriented methods and up and running doing C++ programming! ~~$19.95~~ **$17.95**

NEW! The Complete AppleScript® Handbook by Danny Goodman is a self-contained kit to customizing and enhancing the Macintosh environment. The disk contains AppleScript 1.1 Runtime, Chang Labs TableServer, and useful, ready-to-run scripts. It also shows the Mac user how to automate many processes – no programming experience necessary. ~~$35.00~~ **$31.50**

NEW! The Complete HyperCard® 2.2 Handbook Fourth Edition by Danny Goodman is the biggest-selling Mac book – new revised and updated for version 2.2. It shows how to build working applications using the latest version of HyperCard and covers text, painting tools, extension commands (XCMDs), scripting in HyperTalk, and more. ~~$35.00~~ **$31.50**

NEW! CompuServe For Dummies™ by Wallace Wang Find out how to shop, play games, join forums, get the latest news, do research, and more on this popular on-line service. ~~$19.95~~ **$17.95**

NEW! The Delphi Internet Start-Up Guide Your Personal Guide to Delphi Internet Services by Steve Lambert and Walt Howe. This book showcases the new graphical Delphi Internet online service. There are over 20 million Internet users worldwide, and Delphi – with over 600 local access numbers across the country – is one of the cheapest and most popular commercial services to provide full Internet access. And now, with Delphi's new graphical front end, anyone with a computer, a modem, and a phone line can get on the Internet. The accompanying CD-ROM contains Delphi's new, easy-to-use, graphical interface software for effortless Internet navigation, popular games, and other Internet tools. The book includes a hands-on, guided tour of Delphi's hottest new services and resources. As well as a special offer from Delphi Internet Services that gives the reader immediate access to the Internet, five hours of free online time, and exclusive special discounts. ~~$40.00~~ **$36.00**

The Elements of E-Mail Style by Brent Heslop and David Angell. Learn the rules of the road in the e-mail age. Concise, easy-to-use format explaining essential e-mail guidelines and rules. It covers style, tone, typography, formatting, politics and etiquette. It also outlines basic rules of composition within the special context of writing e-mail and includes samples and templates for writing specific types of e-mail correspondence. 208 pages. ~~$14.95~~ **$13.45**

E-Mail Essentials by Ed Tittel & Margaret Robbins is a hands-on guide to the basics of e-mail, the ubiquitous networks communication system. The book is suitable for both the casual e-mailer and the networking professional, as it covers everything from the installation of e-mail to the maintenance and management of e-mail hubs and message servers. The books explains the fundamental concepts and technologies of electronic mail, featuring chapters on Lotus applications and CompuServe, as well as information on upgrading, automation, message-based applications, and user training. E-mail Essentials is a step-by-step, jargon-free guide that will enable the e-mail user to get the most out of the communication potentials of networking. 250 pp. ~~$24.95~~ **$22.45**

NEW! Danny Goodman's AppleScript Handbook Second Edition by Danny Goodman is a self-contained kit shows the reader how to customize and extend the capabilities of any Macintosh computer – no programming experience needed! This enhanced and expanded edition of The Complete AppleScript Handbook focuses on putting AppleScript to work in all sorts of practical situations. In addition, Danny Goodman's AppleScript Handbook, Second Edition shows you how to apply the same principles to other popular scripting systems, such as Userland Frontier and QuickKeys. Shows readers how to use scripts to enhance the Macintosh environment, automate many processes, link data between applications, and much more. This book provides a wealth of all-new examples showing how to integrate AppleScript with the Finder, spreadsheets, desktop publishing programs, graphics applications, databases, telecommunications programs, utilities, and HyperCard. The accompanying 3 1/2" disk is jam-packed with over $100 worth of software, including AppleScript 1.1, valuable utilities, and powerful, ready-to-use scripts. ~~$39.00~~ **$35.00**

NEW! Danny Goodman's Macintosh® Handbook Featuring System 7 by Danny Goodman with Richard Saul Wurman. This user friendly design includes a unique four-color design and exploded diagrams. It includes over 100 spreads break down and clarify Mac problems and includes insider's tips. ~~$29.95~~ **$26.95**

Graphics Gems IV edited by Paul Heckbert Volume IV is the newest collection of carefully crafted, innovative gems. All of the gems are immediately accessible and useful in formulating clean, fast, and elegant programs. The C programming language is used for most of the program listings, although several of the gems have C++ implementations. An IBM or Macintosh disk containing all of the code from all four volumes is included. Includes one 3.5" high-density disk. ~~$49.95~~ **$44.95**

NEW! Graphics Gems V Edited by Alan W. Paeth is Graphics Gems V is the newest volume in The Graphics Gems Series. It is intended to provide the graphics community with a set of practical tools for implementing new ideas and techniques, and to offer working solutions to real programming problems. These tools are written by a wide variety of graphics programmers from industry, academia, and research. The books in this series have become essential, time-saving tools for many programmers. It is the latest collection of graphics tips in The Graphics Gems Series written by the leading programmers in the field. It contains about 50 new gems displaying the most recent and innovative techniques in graphics programming. Also included is new gems in ellipses, splines, Bezier curves, and ray tracing. Includes a disk which contains source code from all five volumes and is available in both IBM and Macintosh versions. CONTENTS: Algebra and Arithmetic. Computational Geometry. Modeling and Transformation. Curves and Surfaces. Ray Tracing and Radiosity. Halftoning and Image Processing. Utilities. ~~$49.95~~ **$44.95**

How To Write Macintosh Software by Scott Knaster is a great source for understanding Macintosh programming techniques. Drawing from his years of experience working with programmers, Scott explains the mysteries and myths of Macintosh programming with wit and humor. The third edition, fully revised and updated, covers System 7 and 32-bit developments, and explores such topics as how and where things are stored in memory; what things in memory can be moved around and when they may be moved; how to debug your applications with MacsBug; how to examine your program's code to learn precisely what's going on when it runs. 448 pgs. ~~$28.95~~ **$26.05**

NEW! HyperTalk® 2.2: The Book Second Edition by Dan Winkler, Scot Kamins, and Jeanne DeVoto is the most complete, authoritative source on HyperTalk 2.2 programming and troubleshooting. It covers each language element of HyperTalk 2.2 (including the odd quirk or bug). ~~$35.00~~ **$31.50**

NEW! Inside the PowerPC by Tom Badgett The PowerPC, which can run both PC and Macintosh software, signals a new trend in personal computers – and Inside The PowerPC covers this new technology as no other book. Whereas the Pentium represents the end of a long line of development, the PowerPC represents the beginning of a new wave of personal computers – and, because of its high speed and low power consumption, it is especially suitable for the hot selling portable computers. It covers the PowerPC in both its Mac-centric and PC-centric versions (either of which can run the other's software in emulation) and shows the reader how to get the most out of a PowerPC-based machine, including how to work with floppy disks, hard drives, memory, video, printers, networking, modems, and multimedia. Also shown is how to upgrade suitable machines to the PowerPC, and explains in easily understood terms the design and performance of the PowerPC. ~~$30.00~~ **$27.00**

The Instant Internet Guide by Brent Heslop and David Angell. An Internet jump-start – how to access, use and navigate global networks. The Instant Internet Guide equips readers with the tools needed to travel the electronic world. The book highlights the most important sources of Internet news and information and explains how to access information on remote systems. It outlines how to use essential Internet utilities and programs and includes a primer on UNIX for the Internet. 224 pages ~~$14.95~~ **$13.45**

NEW! The Internet by Paul Hoffman gives the straight scoop on the Net with the elegant, entertaining 4-color companion to the PBS special "The Internet Show." Color photos and illustrations explain Internet key features, its history, and trivia. Covers the basics and also a multitude of Internet Information Services and extensive lists of Internet sites. ~~$24.00~~ **$22.50**

NEW! The Internet, Deluxe Edition by Paul Hoffman. All of the brilliance of The Internet plus the hot NetManage Internet Chameleon software! This bundled Deluxe Edition includes everything you need to surf the Net! Fully automated, NetManage has put together 3 disks filled with Internet tools, service providers, applications, and more – an unbelievable value! ~~$34.00~~ **$31.50**

NEW! The Internet For Macs For Dummies™ Starter Kit by Charles Seiter. Access and

Want more product info? Call us at 310/575-4343.

navigate the Net like a pro the fun and easy way with this all-inclusive Starter Kit for Mac users. Kit includes a special edition of the bestselling The Internet For Macs For Dummies™, plus two disks featuring The Pipeline and TCP/Connect II® Extended software with: E-Mail, UseNet News Reader, FTP, Telnet, Gopher, and more! ~~$29.99~~ **$26.99**

NEW! The Internet For Macs For Dummies™ by Charles Seiter. Now novice Mac users have an Internet guide written especially for them – in the easy ...For Dummies™ style they love. ~~$19.99~~ **$17.95**

NEW! The Internet For Dummies™ 2nd Edition by John Levine and Carol Baroudi. Surf the net with ease using the up-to-the-minute new edition of the #1 bestselling Internet reference – now with friendly help on connecting to the Internet. ~~$19.99~~ **$17.99**

NEW! MORE Internet For Dummies™ by John Levine & Margaret Levine Young. The expanded guide full of great Internet tips for all those users who want to know where to go and what to do once connected. ~~$19.95~~ **$17.95**

NEW! The Internet For Dummies™ Quick Reference by John Levine. This fact-filled quick reference provides plain English explanations of Internet terms and basics. Cross-referenced to The Internet For Dummies. ~~$8.95~~ **$8.05**

NEW! Internet Power Tools by John Ross is designed for intermediate PC users, Internet Power Tools is a complete book/disk package that allows access to the world's largest network – the Internet – with the same ease as one uses Windows. The detailed instructions allow readers to find their way around the Internet fast and covers e-mail, file transfers, remote logon, on-line directories, and more. The accompanying disk is packed with powerful utilities, including Cello, PC Eudora, and Panda – easy-to-use graphical user interfaces (GUI) that obviate the need for the obscure Unix line commands that everyone else must learn. It also provides access to even more software via Random House's Internet FTP site and shows how to connect to the Internet via a network connection or dial-in. ~~$40.00~~ **$36.00**

NEW! The 1994 Internet White Pages by Seth Godin & James S. McBride is the one and only complete alphabetical directory of people on the Internet. ~~$29.95~~ **$26.95**

NEW! Internet SECRETS by John Levine & Carol Baroudi gives the most from the Net with this performance-oriented book – for Windows, UNIX, DOS, and Mac Internet users who know how to get connected but want to optimize their connections. ~~$39.99~~ **$35.99**

Learn C on the Macintosh by Dave Mark. This self-teaching book/disk package gives you everything you need to begin programming on the Macintosh. Learn to write, edit, compile, and run your first C programs through a series of over 25 projects that build on one another. The book comes with THIN C – a customized version of Symantec's THINK C, the leading programming environment for Macintosh. 464 pages, Book/disk: ~~$34.95~~ **$31.45**

NEW! Mac Programming for Dummies™ by Dan Parks Sydow takes the intimidation and work out of writing Mac programs. ~~$19.95~~ **$17.95**

NEW! Mac Screamer The Ultimate Macintosh® Supercharging Kit by Jan Harrington covers 30 Macintosh models, including the Classics, LCs, PowerBooks, and Quadras and gives software solutions and hardware tips to accelerate Mac performance. It lets readers in on do-it-yourself tips that can save them over 25% on upgrade costs. ~~$35.00~~ **$31.50**

NEW! Macintosh® Crash Course by Glenn Brown shows Macintosh power users what to do when things go wrong with their system. Macintosh Crash Course shows readers how to overcome Macintosh system crashes, system lock-ups, and various, frustrating and cryptic error messages they regularly encounter. It includes a CD-ROM with shareware and freeware to help the user diagnose and repair system failures. Includes up-to date coverage through Macintosh System 7.5, Managing memory, Hardware diagnostics, File recovery, PowerBook problems, PowerPC problems, network utilities, hard drive repair utilities, SCSI problems, conflicts and solutions and File synchronization and utilities. ~~$29.95~~ **$26.95**

Learn C++ on the Macintosh by Dave Mark. After a brief refresher course in C, Learn C++ introduces the basic syntax of C++ and object programming. Then you'll learn how to write, edit, and compile your first C++ programs through a series of programming projects that build on one another as new concepts are introduced. Key C++ concepts such as derived classes, operator overloading, and iostream functions are all covered in Dave's easy-to-follow approach. Includes a special version of Symantec C++ for Macintosh. Book/disk package with 3.5" 800K Macintosh disk. 400 pages, ~~$36.95~~ **$33.26**

Macintosh C Programming Primer Volume I, Second Edition, Inside the Toolbox Using THINK C by Dave Mark and Cartwright Reed. This new edition of this Macintosh programming bestseller is updated to include recent changes in Macintosh technology, including System 7, new versions of THINK C and ResEdit, and new Macintosh machines. Readers will learn how to use the resources, Macintosh Toolbox and interface to create stand-alone applications. 672 pages, ~~$26.95~~ **$24.25**

Macintosh C Programming Primer Volume II, Mastering the Toolbox Using THINK C by Dave Mark. Volume II picks up where Volume I leaves off, covering more advanced topics such as: Color QuickDraw, THINK Class Library, TextEdit, and the Memory Manager: 528 pgs. ~~$26.95~~ **$24.25**

Macintosh Pascal Programming Primer Volume I, Inside the Toolbox Using THINK Pascal by Dave Mark and Cartwright Reed. This tutorial shows programmers new to the Macintosh how to use the Toolbox, resources, and the Macintosh interface to create stand-alone applications with Symantec's THINK Pascal. 544 pages ~~$26.95~~ **$24.25**

Macintosh Programming Techniques by Dan Sydow (Series Editor: Tony Meadow). This tutorial and handbook provides a thorough foundation in the special techniques of Macintosh programming for experienced Macintosh programmers as well as those making the transition from DOS, Windows, VAX or UNIX. Emphasizes programming techniques over syntax for better code, regardless of language. Guides the reader through Macintosh memory management, QuickDraw, events and more, using sample program in C++. Disk includes an interactive tutorial, plus reusable C++ code. ~~$34.95~~ **$31.95**

NEW! Macworld Ultimate Mac Programming by Dave Mark Bestselling Mac programming author Dave Mark reveals the secrets of Mac programming and presents important, timesaving techniques. ~~$39.95~~ **$35.95**

NEW! Mosaic For Dummies™, Windows Edition by David Angell & Brent Helsop. Learn to use Mosaic with the only book that explains the most popular viewer of the World Wide Web in plain English. Covers the popular Windows version of Mosaic. ~~$19.99~~ **$17.99**

Multimedia Authoring: Building and Developing Documents by Scott Fisher addresses the concerns that face anyone trying to create multimedia documents. It offers specific advice on when to use different kinds of information architecture, discusses the human-factors concepts that determine how readers use and retain information, and then applies these findings to multimedia documents, covering the high-level issues concerning planners and authors of multimedia documents as well as those involved in evaluating or purchasing multimedia platforms. Includes one 3.5" high-density disk. ~~$34.95~~ **$31.45**

NEW! Net Chat™ by Michael Wolff & Co. There are an estimated 35 million people online and most of them spend time chatting on the Net. Net Chat is the first comprehensive, discriminating guide devoted to the vast world of online interpersonal communications. Following the successful format of the ever popular Net Guide, Net Chat is designed for easy access, featuring an attractive layout, informative illustrations, and thousands of choices. It is encyclopedic in scope – covers "chat" groups on the Internet, all the major commercial services (including CompuServe, Prodigy, and America Online), and hundreds of bulletin boards. It features a map of salons and meeting places in Cyberspace, where people stop to chat about politics, lifestyle, music, dating, sex, fantasy, health, family, and just about everything imaginable! The entries include one- and two-page spreads that describe the featured topic, and provide easy-access instructions, complete listings, and visual images from the Net. For use with all computer platforms – PCs, Macs, workstations. ~~$19.99~~ **$17.00**

NEW! PowerBook™ The Digital Nomad's Guide by Andrew Gore and Mitch Ratcliffe. ~~$24.00~~ **$21.60**

Programming for the Newton Software Development with NewtonScript by Julie McKeehan and Neil Rhodes. Foreword by Walter R. Smith. Programming for the Newton: Software Development with NewtonScript is an indispensable tool for Newton programmers. Readers will learn how to develop software for the Newton on the Macintosh from people that developed the course on programming the Newton for Apple Computer. The enclosed 3.5" disk contains a sample Newton application from the books, as well as demonstration version of Newton Toolkit (NTK), Apple Computers complete development environment for the Newtons. A Publication of AP Professional May 1994, Paperback, 393 pp. ~~$29.95~~ **$26.95**

Programming in Symantec C++ for the Macintosh by Judy May and John Whittle. This book will introduce you to object-oriented programming, the C++ language, and of course Symantec C++ for the Macintosh. You don't have to be a programmer, or even know anything about programming to benefit from this book. Programming in Symantec C++ for the Macintosh covers everything from the basics to advanced features of Symantec C++. If you are a Think C or Zortech C++ programmer who wants to learn more about object-oriented programming or what's different about Symantec C++, there are chapters specifically for you. Includes helpful examples of C++ code that illustrate object-oriented programs. ~~$29.95~~ **$26.95**

Programming for System 7 by Gary Little and Tim Swihart, is a hands-on guide to creating applications for

Want more product info? E-mail us at productinfo@xplain.com

System 7. It describes the new features and functions of the operating system in detail. Topics covered include file operations, cooperative multitasking, Balloon Help, Apple events, and the File Manager. Numerous working C code examples show programmers how to take advantage of each of these features and use them in developing their applications. 384 pages. ~~$26.95~~ **$24.25**

ResEdit™ Complete, Second Edition by Peter Alley and Carolyn Strange. With ResEdit, Macintosh programmers can customize every aspect of their interface form creating screen backgrounds and icons to customizing menus and dialog boxes. 608 pages. Book/disk package. ~~$34.95~~ **$31.45**

NEW! Programming Primer For The Macintosh® Volume 1 by John Whittle and Judy May. This book provides an introduction to Macintosh programming, using C++ as the example language, and provides realistic, easy to follow, programming examples designed to work with either Symantec® C++ or Metrowerks® CodeWarrior™. Also includes one 3.5" disk with source code for the programming examples, along with numerous, useful, public domain utilities to use with each compiler. ~~$37.95~~ **$34.15**

Sad Macs, Bombs, Disasters and What to Do About Them by Ted Landau comes to the rescue with your Macintosh problems. From fractious fonts to the ominous Sad Macintosh icon, this emergency handbook covers the whole range of Macintosh problems: symptoms, causes, and what you can do to solve them. 640 Pages. ~~$24.95~~ **$22.45**

Software By Design: Creating User Friendly Software by Penny Bauersfeld (Series Editor: Tony Meadow). This excellent reference provides readers with a thorough how-to for designing software that is easy to learn, comfortable to operate and that inspires user confidence. Written from the perspective of Macintosh, but compatible with all platforms. Stresses user input from initial design, through prototyping, testing and revision. Provides tools for analyzing user needs and test responses. Includes exercises for sharpening user-oriented design skills. ~~$29.95~~ **$26.95**

Taligent's Guide to Designing Programs: Well-Mannered Object-Oriented Design in C++ is the Taligent approach to object-oriented design. The Taligent Operating Environment is the first commercial software system based entirely on object-oriented technology. Taligent's Guide to Designing Programs is a developer's-eye view of this system. It introduces new concepts of programming and empowers developers to create software more productively. Out of their direct experience in developing the system, the authors focus on global issues of object-oriented design and writing C++ programs, and the specific issues of programming in the Taligent Operating Environment. Taligent's Guide to Designing Programs assumes the reader is an experienced C++ programmer, and proceeds from there to fully explore "the Taligent way" of programming. ~~$19.50~~ **$17.55**

NEW! Wireless For The Newton Software Development for Mobile Communications by Julie McKeehan and Neil Rhodes is a book that picks up where Programming for the Newton left off, teaching the reader how to develop Newton® software on the Macintosh. The enclosed floppy disk provides a sample application, as well as a fully functional demonstration version of Newton Toolkit™ (NTK™), Apple Computer's complete development environment for the Newton®. Gives hands-on Newton environment training with sample code created specifically for the Newton®. The authors are external faculty at Apple Developer University teaching classes on programming for the Newton®. Programming experience is assumed, although not in any particular language. Enclosed is a floppy disk which contains source code for a Newton application, as well as demonstration NTK™. ~~$34.95~~ **$31.45**

Writing Localizable Software for the Macintosh by Daniel R. Carter. 469 pages. ~~$26.95~~ **$24.25**

THE APPLE LIBRARY

NEW! Macintosh Programmer's Toolbox Assistant CD-ROM Instant electronic access to Inside Macintosh essentials. Now Macintosh programmers can get quick access to over 4,000 Toolbox calls that are at the heart of Macintosh system software. The definitions of these data structures, resources, constants, and functions are documented in the Inside Macintosh series and are essential information for anyone developing Macintosh software. Macintosh Programmer's Toolbox Assistant is a CD-ROM that harnesses the power of one of the best search and viewing engines in the industry. It allows programmers to access the Toolbox calls quickly from their development environment. With hypertext links allowing programmers to view related topics easily. Macintosh Programmer's Toolbox Assistant is the ultimate electronic reference tool for Macintosh programmers. $99.95

HyperCard Stack Design Guidelines by Apple Computer, Inc. is an essential book for everyone who creates Apple HyperCard stacks, from beginners to commercial developers. It covers the basic principles of design that, when incorporated, make HyperCard stacks effective and usable. Topics include guidelines, navigation, graphic design and screen illustration, text in stacks, music and sound, a sample stack development scenario, collaborative development, and the Stack Design Checklist. 240 pages. ~~$21.95~~ **$19.95**

Inside AppleTalk by Gursharan S. Sidhu, Richard F. Andrews and Alan B. Oppenheimer. Apple Computer, Inc. 650 pages. ~~$34.95~~ **$31.45**

Inside Macintosh: AOCE Application Interfaces by Apple Computer, Inc. shows how your application can take advantage of the system software features provided by PowerTalk system software and the PowerShare collaboration servers. Nearly every Macintosh application program can benefit from the addition of some of these features. This book shows how you can add electronic mail capabilities to your application, write a messaging application or agent, store information in and retrieve information from PowerShare and other AOCE catalogs, add catalog-browsing and find-in-catalog capabilities to your application, write templates that extend the Finder's ability to display information in PowerShare and other AOCE catalogs, add digital signatures to files or to any portion of a document, and establish an authenticated messaging connection. ~~$40.45~~ **$36.40**

Inside Macintosh: AOCE Service Access Modules by Apple Computer, Inc. describes how to write a software module that gives users and PowerTalk-enabled applications access to a new or existing mail and messaging service or catalog service. This book shows how to write a catalog service access module (CSAM), a messaging service access module (MSAM), and AOCE templates that allow a user to set up a CSAM or MSAM and add addresses to mail and messages. ~~$26.95~~ **$24.25**

NEW! Inside Macintosh: CD-ROM by Apple Computer, Inc. Inside Macintosh® is the essential reference for programmers, designers, and engineers for creating applications for the Macintosh family of computers. Inside Macintosh CD-ROM collects more than 25 volumes in electronic form, including: QuickDraw™ GX Library, Macintosh Human Interface Guidelines, PowerPC System Software, Macintosh Toolbox Essentials and More Macintosh Toolbox, QuickTime and QuickTime Components. Now programmers will be able to access over 16,000 pages of the information they need directly from their computers. Hypertext linking and extensive cross referencing across volumes allows programmers to search and explore this library in ways that are unique to the electronic medium. Every Macintosh programmer will regard Inside Macintosh CD-ROM as their most important resource. $99.95

Inside Macintosh: Devices by Apple Computer, Inc. describes how to write software that interacts with built-in and peripheral hardware devices. With this book,. you'll learn how to write and install your own device drivers, desk accessories, and Chooser extensions; communicate with device drivers using the Device Manager; access expansion cards using the Slot Manager; control SCSI devices using SCSI Manager 4.3 or the original SCSI Manager; communicate directly with Apple Desktop Bus devices; interact with the Power Manager in battery-powered Macintosh computers; and communicate with serial devices using the Serial Driver. ~~$29.95~~ **$26.95**

Inside Macintosh: Files by Apple Computer, Inc. describes the parts of the operating system that allow you to manage files. It shows how your application can handle the commands typically found in a File menu. It also provides a reference to the File and Alias Managers, the Disk Initialization and Standard File Packages. 510 pgs. ~~$29.95~~ **$26.95**

Inside Macintosh: Interapplication Communication by Apple Computer, Inc. shows how applications can work together. How your application can share data, request information or services, allow the user to automate tasks, communicate with remote databases. ~~$34.95~~ **$31.45**

Inside Macintosh: Imaging by Apple Computer, Inc. covers QuickDraw and Color QuickDraw. The book includes general discussions of drawing and working with color. It describes the structures that hold images and image information, and the routines that manipulate them. It also covers the Palette, Color, and Printing Managers, and the Color Picker, Color Matching, and Picture Utilities. ~~$26.95~~ **$24.25**

Inside Macintosh: Macintosh Toolbox Essentials by Apple Computer, Inc. covers the heart of the Macintosh. The toolbox enables programmers to create applications consistent with the Macintosh "look and feel". This book describes Toolbox routines and shows how to implement essential user interface elements, such as menus, windows, scroll bars, icons and dialog boxes. 880 pages. ~~$34.95~~ **$31.45**

Inside Macintosh: More Macintosh Toolbox by Apple Computer, Inc. covers other Macintosh features such as how to support copy and paste, provide Balloon Help, play and record sound and create control panels are covered in this volume. The managers discussed include Help, List, Resource, Scrap and Sound. ~~$34.65~~ **$31.45**

Inside Macintosh: Memory by Apple Computer, Inc. describes the parts of the Macintosh operating

Want more product info? Call us at 310/575-4343.

system that allow you to manage memory. It provides detailed strategies for allocating and releasing memory, avoiding low-memory situations, reference to the Memory Manager, the Virtual Memory Manager, and memory-related utilities. 296 pages. ~~$24.95~~ **$22.45**

Inside Macintosh: Networking by Apple Computer, Inc. describes how to write software that uses AppleTalk networking protocols. It describes the components and organization of AppleTalk and how to select an AppleTalk protocol. It provides the complete application interfaces to all AppleTalk protocols, including ATP (AppleTalk Transaction Protocol), DDP (Datagram Delivery Protocol), and ADSP (AppleTalk Data Stream Protocol), among others. ~~$29.95~~ **$26.95**

Inside Macintosh: Operating System Utilities by Apple Computer, Inc. describes parts of the Macintosh Operating System that allow you to manage various low-level aspects of the operating system. Everyone who programs the Macintosh should read this book! It will show you in detail how to get information about the operating system, manage operating system queues, handle dates and times, control the settings of the parameter RAM, manipulate the trap dispatch table, and receive and respond to low-level system errors. ~~$26.05~~ **$23.45**

Inside Macintosh: Overview by Apple Computer, Inc. is the first book that people who are unfamiliar with Macintosh programming should read. It gives an overview of Macintosh programming fundamentals and a road map to the New Inside Macintosh library. Inside Macintosh: Overview also covers various programming tools and languages, compatibility guidelines and an overview of considerations for worldwide development. 176 pages. ~~$22.95~~ **$20.65**

Inside Macintosh: PowerPC Numerics by Apple Computer, Inc. describes the floating-point numerics environment provided with the first release of PowerPC processor-based Macintosh computers. The numerics environment conforms to the IEEE standard 754 for binary floating-point arithmetic. This book provides a description of that standard and shows how RISC Numerics compiles with it. This book also shows programmers how to create floating-point values and how to perform operations on floating-point values in high-level languages such as C and in PowerPC assembly language. ~~$28.95~~ **$26.00**

Inside Macintosh: PowerPC System Software by Apple Computer, Inc. describes the new process execution environment and system software services provided with the first version of the system software for Macintosh on PowerPC computers. It contains information you need to know to write applications and other software that can run on the PowerPC. PowerPC System Software shows in detail how to make your software compatible with the new run-time environment provided on PowerPC-based Macintosh computers. It also provides a complete technical reference for the Mixed Mode Manager, the Code Fragment Manager, and the Exception Manager. ~~$24.95~~ **$22.45**

Inside Macintosh: Processes by Apple Computer, Inc. describes the parts of the Macintosh operating system that allow you to control the execution of processes and interrupt tasks. It shows in detail how you can use the Process Manager to get information about processes loaded in memory. It is also a reference for the Vertical Retrace, Time, Notification, Deferred Task, and Shutdown Managers. 208 pages. ~~$22.95~~ **$20.65**

Inside Macintosh: QuickTime by Apple Computer, Inc. is for anyone who wants to create applications that use QuickTime, the system software that allows the integration of video, animation, and sounds into applications. This book describes all of the QuickTime Toolbox utilities. In addition, it provides the information you need to compress and decompress images and image sequences. ~~$29.95~~ **$26.95**

Inside Macintosh: QuickTime Components by Apple Computer, Inc.covers how to use and develop QuickTime components such as image compressors, movie controllers, sequence grabbers, and video digitizers. ~~$34.95~~ **$31.45**

Inside Macintosh: Sound by Apple Computer, Inc. describes the parts of the Macintosh system software that allow you to manage sounds. It contains information that you need to know to write applications and other software that can record and play back sounds, compress and expand audio data, convert text to speech, and perform other similar operations. ~~$26.95~~ **$24.25**

Inside Macintosh: Text by Apple Computer, Inc. describes how to perform text handling, from simple character display to multi-language processing. The Font, Script, Text Services, and Dictionary Managers are all covered, in addition to QuickDraw Text, TextEdit, and International and Keyboard Resources. ~~$39.95~~ **$35.95**

Inside Macintosh: QuickDraw™ GX Library by Apple Computer Inc. is the powerful new graphics architecture for the Macintosh. Far more than just a revision of QuickDraw, QuickDraw GX is a unified approach to graphics and typography that gives programmers unprecedented flexibility and power in drawing and printing all kinds of shapes, images, and text.

Inside Macintosh: QuickDraw GX Objects by Apple Computer, Inc. introduces QuickDraw GX and its object structure, and shows programmers how to manipulate objects in all types of programs. ~~$26.95~~ **$24.25**

Inside Macintosh: QuickDraw GX Graphics by Apple Computer, Inc. shows readers how to create and manipulate the fundamental geometric shapes of QuickDraw GX to generate a vast range of graphic entities. It also demonstrates how to work with bitmaps and pictures, and specialized QuickDraw GX graphic shapes. ~~$26.95~~ **$24.25**

NEW! Inside Macintosh®: QuickDraw™ GX Environment and Utilities A companion to QuickDraw™ GX Objects, this book contains programming information useful to any developer writing QuickDraw GX applications. It describes QuickDraw GX memory management, error handling, debugging, and mathematical functions, as well as conversion from QuickDraw to QuickDraw GX. ~~$29.95~~ **$26.95**

NEW! Inside Macintosh®: QuickDraw™ GX Printing This book is essential for any developer whose QuickDraw™ GX application supports printing. It shows how to support the new printing features of QuickDraw GX, including desktop printers and expandable printing dialog boxes. QuickDraw GX Printing also shows how to use printing-related objects to add custom panels to printing dialog boxes and to create custom page formats. ~~$26.95~~ **$24.25**

NEW! Inside Macintosh®: QuickDraw™ GX Printing Extensions and Drivers Any developer who wants to create extensions to the application printing capabilities of QuickDraw™ GX, or who needs to write a printing device driver that works with QuickDraw GX needs this book. QuickDraw GX Printing Extensions and Drivers describes how to create printing extensions and printer drivers, and provides a complete reference to the messages, functions, and resources that they use. ~~$29.95~~ **$26.95**

NEW! Inside Macintosh®: QuickDraw™ GX Programmer's Overview This book provides an introduction to QuickDraw™ GX, providing an overview of the QuickDraw GX environment from a developer's perspective. It introduces the QuickDraw™ GX programming and runtime environments, the relationship between QuickDraw GX and the rest of the Macintosh® systems software and the relationship between QuickDraw GX and Macintosh applications. The key elements of QuickDraw GX programming, data structures, object types, and functions used most frequently by QuickDraw GX developers are also covered. After a general introduction, this book provides readers with a series of practical examples demonstrating how to approach programming with QuickDraw GX. ~~$24.95~~ **$22.45**

NEW! Inside Macintosh®: QuickDraw™ GX Typography This book is essential for any developer who uses QuickDraw™ GX to manipulate text. It shows how to use QuickDraw GX objects to handle all kinds of text – from plain, unstyled text to complex, mixed-direction and multi-language text with sophisticated stylistic and typographic variations. QuickDraw GX Typography shows how to create and manipulate the three different types of text shapes supported by QuickDraw GX including text shapes, glyph shapes, and layout shapes. ~~$29.95~~ **$26.95**

NEW! Inside Macintosh: X-Ref. by Apple Computer, Inc. is an index for Inside Mac. ~~$12.95~~ **$11.65**

LANGUAGES

CodeWarrior™ CD by Metrowerks comes in two versions – Bronze and Gold. These CDs contain the CodeWarrior development environment including C++, C and Pascal compilers; high-speed linkers; native-mode interactive debuggers; and a powerful new application framework called PowerPlant for rapid Macintosh development in C++. Bronze generates 680x0 code. Gold generates both 680x0 and PowerPC code. All versions are a 3 CD subscription over a 1-year period. Bronze: $99, Gold: $399. **Bronze comes with a 6-month MacTech subscription. Gold comes with a 1-year subscription. Both at no additional charge!**

NEW! Geekware by Metrowerks is here! In high school, they called you a computer geek. Now, they work at burger joints and wear polyester uniforms. And you don't. Wear it to your favorite burger joint. $24.95

BASIC for the Newton is BASIC for the Newton! From NS BASIC Corporation, it is a fully interactive implementation of the BASIC programming language. It runs entirely on the Newton – no host is required. It includes a full set of functions and data types, hand-written input, windows, buttons and extensions to take advantage of the Newton environment. Applications can create files or access the built-in soups. Applications can also access the serial port for input and output. Work directly on the Newton, or through a connected Mac/PC and keyboard. NS BASIC includes a

Want more product info? E-mail us at productinfo@xplain.com

MAIL ORDER STORE

150 page pocket sized manual. $99

SmalltalkAgents™, a superset of the Smalltalk language, is fully integrated with Macintosh, incorporating design features specifically for the RISC and Macintosh System 7 architecture. SmalltalkAgents is a true object oriented workbench that includes an incremental and extensible compiler, an array of design and cross reference tools, preemptive interrupt driven threads and events, an extensive class library including classes for general programming, classes for the Macintosh user interface and classes for the Macintosh operating system. Integration of components in enterprise systems is simplified with the network, telecommunication, and inter-application communication libraries. The SmalltalkAgents' extensive class library and add-on components make it especially well suited as a development workbench for custom applications in business, education, science, engineering, and academic research. $695

Symantec C++ for Macintosh is an object oriented development environment designed for SYMANTEC.™ professional Macintosh programmers. Symantec C++ features powerful object-oriented development tools within a completely integrated environment. The C++ compiler, incremental linker, THINK Class Library, integrated browser, and automatic project management give Symantec C++ fast turnaround times. This product supports multiple editors and translators, so you can use your favorite tools and resource editors as well as scripts you've written within the environment. And with ToolServer, you'll be able to customize menus and attach scripts based on Apple events, AppleScript, and MPW Tools. The built-in SourceServer provides a source code control system, allowing teams of programmers to solve tough problems faster. With SourceServer, you'll always know you're working on the latest version. And you'll have old versions at your fingertips when code "breaks" and you need to look back at modifications. Product Contents: Three high density disks, an 832-page user manual, a 568-page THINK Class Library and a 100-page C++ Compiler Guide. $369

THINK C by Symantec Corporation. THINK C is easy to use and highly visual, making it the No. 1 selling Macintosh programming environment. Enhancements make this product faster and more versatile than ever, improving your productivity with more powerful project management, a full set of tools, and script support for major script-based languages. With the THINK environment, you spend less time on routine programming tasks due to an extremely fast compiler and incremental linker. In addition, the automatic project manager saves you time by tracking changes to your files and recompiling only those that have changes. All the tools you need – a multi-window editor, compiler, linker, debugger, browser, and resource editor – are completely integrated for speed and ease of use. One of the most valuable of these tools is the THINK Class Library, a set of program building blocks that gives you a head start in writing object-oriented applications. And with the new open architecture, you can use your favorite tools, resource editors, and scripts within the environment. THINK C is the logical next step for programmers who have worked in HyperCard or other script-based development environments. The environment supports AppleScript, Apple events, and Frontier, so you can link and automate complex, multi-project operations. Product Contents: Four Macintosh disks, an 832-page user manual, and a 568-page THINK Class Library Guide. $219

THINK Pascal v. 4.0 by Symantec Corporation. Professionals and students will welcome this version of THINK Pascal. It is fully integrated for rapid turnaround time and lets you take advantage of System 7 capabilities. Features include support for large projects, enhanced THINK Class Library, System 7 compatibility, superior code generation, and smart linking. Product Contents: Four Macintosh disks, a 562-page user manual, and a 498-page object-oriented programming manual. $169

UTILITIES

BBEdit 3.1 from Bare Bones Software is now better than ever. In addition to being Accelerated for Power Macintosh, this powerful, intuitive text editor offers integrated support for THINK C 7.0, Metrowerks CodeWarrior, THINK Reference 2.0 and MPW ToolServer. Version 3.1 adds even more capability, including "soft" wrapping of text on screen and numerous refinements and improvements to the user interface. BBEdit's many features include: Integrated PopupFuncs™ technology for speedy navigation of source code files (C, C++, Pascal, Rez, 68K Assembler, and Fortran), unique 'Find Differences' command (BBEdit can find differences between projects and folders as well as files), support for Macintosh Drag and Drop for editing and other common tasks, PowerTalk support for reading, sending and composition of PowerTalk mail, scripting via any OSA compatible scripting language including AppleScript and Frontier 3.0, and fast search and replace with optional "grep" matching and multi-file searching. BBEdit's robust feature set and proven performance and reliability make it the editor of choice for professionals and hobbyists alike. $119

NEW! **Call Tree** by Barking Dog Software Co is a quick way to understand the structure of complex programs. Utilizing Drag & Drop, the user can produce a call tree analysis of C source code files to assist in optimizing, structuring and a general understanding of software systems. Call Tree is stand-alone and works with CodeWarrior, Think C, Symantec, MPW and most other C source text files. Call Tree is simple to use and can save many hours of tedious work. $149.

CLImate by Orchard Software is a command line interface that lets you communicate with your Macintosh using English commands to create, delete, rename, and move files and folders. It can start applications, format disks, restart your computer, etc. CLImate supplements the Finder. It includes a BASIC interpreter that lets you script your Macintosh without AppleScript. The interpreter includes advanced programming constructs: repeat loops, if/then/else conditionals, subroutine calls, etc... CLImate implements wildcard characters, enabling you to work on groups of files. Use CLImate instead of MPW to manage your projects. CLImate is an application occupying 70K disk space. It comes bundled with sample programs and full documentation. $59.95

CMaster 2.0 by Jersey Scientific installs into THINK C 5 / 6 / 7 and Symantec C++ for Macintosh, and enhances the editor. Use its function popup to select a function and CMaster takes you right to it. Other features include multiple clipboards and markers, a Function Prototyper, and a GoBack Menu which can take you back to previous editing contexts. Almost all features bindable to the keyboard, along over a hundred keyboard-only features like "Add New Automatic Variable." Glossaries, AppleScript and ToolServer support, Macros, and External Tools you create too! $129.95

Cron Manager by Orchard Software implements the UNIX Cron facility. It can open any Macintosh file on a given date and time. By creating an alias, renaming it to the date and time to open, and moving it into the special Cron Events Folder, Cron Manager will open it. Cron Manager is a control panel that creates the special Cron Events Folder inside your System Folder. It is completely transparent to the user. It works like the Startup Items folder, only smarter. It works with any Macintosh file: if you can double-click to start it, Cron Manager can open it. $26.95. Cron Manager bundled with CLImate, $59.95

dtF is a true relational database system for Apple Macintosh computers. dtF provides a powerful choice for developers who want to create database centered applications with no performance trade-offs. dtF features SQL, full transaction control, error recovery, single user, client server architecture and multi-platform support including DOS, Windows, OS/2 and UNIX. The C/C++ API is identical and fully portable cross all supported platforms. Third-party vendors supporting dtF will be able to offer a variety of advanced features and benefits to their customers royalty free. Tools are included for importing, exporting, creating and managing databases and users. Supported development environments include: Symantec, MPW, MetroWerks and more. Mac/SDK: $695

InstallerPack™ by StepUp Software is a package of several Installer "atoms" that let developers incorporate graphics, sounds, file compression and custom folder icons into installation scripts. Compression formats supported are Compact Pro & Diamond. Each atom also available separately: $219

Last Resort Programmer's Edition records every keystroke, command key and mouse event (in local coordinates) to a file on your hard disk. This is especially useful for program testing & debugging, and for technical support and help desks. If something goes wrong (because of a power failure, system crash, forgetting to save or deleting lines) and you lose a word, phrase, or document you can look in the Last Resort keystroke file and recover what you typed. Last Resort is also useful for technical support personnel, when they have to ask "What was the last thing you did before...?" $74.95

NEW! **LJ Profiler** by Lars Jordebo Datakonsult supports profiling of C++ 68K and Power PC applications compiled with CodeWarrior, CFront or Symantec C++. Based on active profiling, i.e. profiling code called at function enter and exit, the browser application lets you follow call chain timings in hierarchical views or separate windows. Collect, organize, compare and save profiling data from different versions of your application into a project. Scriptable and recordable with full access to most internal data structures. Optional remote profiling and tracking of segment and stack usage. Full source code to what you link into your application. $295.

LS Object Pascal CD includes the world's first Object Pascal compiler for Power Macintosh. 100% compatible with Apple's MPW Pascal, LS Object Pascal combines the best of Apple's native development tools with innovative new technology developed at Language Systems. Compiler options specify 68K or native PowerPC code generation. Included on the CD are: LS Object Pascal compiler, Universal Pascal Toolbox interfaces, fully loaded

Want more product info? Call us at 310/575-4343.

MPW 3.3.1, 68K and PowerPC source debuggers, PowerPC assembler, online documentation, Macintosh Tech Notes, and a special version of AppMaker by Bowers Development that generates native Pascal source code. The beta release includes upgrades to v1.0 when it becomes available. $399

Spellswell 7 1.0.4 is an award-winning, comprehensive, practical spelling checker that works in batch mode or within applications that incorporate the Apple Events Word Services protocol (e.g., Eudora, WordPerfect, Communicate!, and Fair Witness). Spellswell 7 checks for spelling errors as well as common typos like capitalization errors, spaces before punctuation, double double word errors, abbreviation errors, mixed case errors, extra spaces between words, a/an before vowel/consonant, etc... MacTech orders include developer kit with Writeswell Jr., a sample Apple Events Word Services word-processor and its source code. $74.95

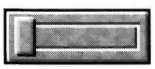
MacAnalyst/Expert, Demo $79, Product $1595 **MacAnalyst** Demo $79, Product $995 **MacDesigner/ Expert,** Demo $79, Product $1595. **MacDesigner** Demo $79, Product $995. By Excel Software. Available. Call for more information about these products.

MacDesigner/Expert by Excel Software supports software engineering methods with the capabilities of MacDesigner plus multi-task design. An integrated requirement database provides traceability from requirement statements to design diagrams, code or test procedures. This tool is well suited to design or maintenance of real-time, multi-tasking software projects.

MacA&D by Excel Software combines the capabilities of MacAnalyst/Expert and MacDesigner/Expert into a single application. It supports structured analysis and design, object-oriented analysis and design, real-time extensions, task design, data modeling, screen prototyping, code editing and browsing, reengineering, requirement traceability, and a global data dictionary. Demo $149, Product $2995

MacWireFrame by Amplified Intelligence. Create your own virtual reality application with MacWireFrame, a virtual reality application frame work. Includes a complete library of object oriented graphics routines, its own easy to understand application frame work (similar to MacApp or TCL but a lot easier to understand), plus an example application program that lets you start solid modeling right away. Comes complete with fully documented source code. All new purchases will be guaranteed a $49.99 upgrade to the soon to be released, scriptable, MacWireFrame 5.0. Due to the overwhelming response the special price offer has been extended for a little while longer. **Special Offer:** ~~$299.00~~ **$75!!!!**

The Memory Mine™ by Adianta is a stand alone debugging tool for Macintosh and native PowerPC. Programmers can monitor heaps, identify problems such as memory leaks, and stress test applications. Active status of memory in a heap is sampled on the fly: allocation in non-relocatable (Ptr), relocatable (Handle) and free space is shown, as are heap corruption, fragmentation, and more... Allocate, Purge, Compact, and Zap memory let users stress test all or part of a program. Source code is not needed to view heaps. It works on Macintoshes with 68020 or later and System 7.0 or later. $99



PictureCDEF 1.3 by Paradigm Software is a professional-level CDEF for creating custom graphical buttons (8-64 pixels). PictureCDEF is used in products by Adobe, ProVue, STF Technologies and others. It is multi-monitor and bit-depth sensitive. The button graphic (cicn, ResEdit) can be changed at runtime and even animated with a call-back routine. Create distinct buttons in seven variations: MultiState, PushButton, FlexiButton, ToggleButton, ChkButton, PushPictButton and TogglePictButton. Position the optional button title at left, bottom or right, or follow the system text direction for international support. Manual, sample code and MacApp 3.0 support included. Full source code: $95.00 Object code: $45.00.

Qd3d/3dPane/SmartPane source code bundle by Vivistar Consulting. **Qd3d 2.0:** Full featured 3d graphics. Points; lines; polygons; polyhedra; Gouraud shading; z-buffering; culling; depth cueing; parallel, perspective, and stereoscopic projections; performance enhancing "OnlyQD" and "Wireframe" modes; full clipping; pipeline access; animation and model interaction support; and a "triad mouse" to map 2d mouse movement to 3d. **3dPane 2.0:** Integrates Qd3d with the TCL and provides a view orientation controller. **SmartPane:** Offscreen image buffering, flicker free animation, and QuickTime movie recording. For use with Qd3d/3dPane or in 2d settings. All work with C++ compilers or THINK C 6. $192

QC™ by Onyx Technology, is a system extension that stress tests code during runtime for common and not-so-common errors. Tests include heap checks, purges, scrambles, handle/pointer validation, dispose/release checks, write to zero, de-reference zero as well as other tests like free memory invalidation and block bounds checking. QC is extremely user friendly for the non-technical tester yet offers an API for programmers who want precise control over testing. QC is Also available in Japanese. $99.95

NEW! **QUED/M 2.7** by Nisus Software, is a programmer's text editor which has defined the industry standard for speed and efficiency. With integrated support for Symantec C/C++, Metrowerks CodeWarrior, and MPW, QUED/M offers unrivaled usefulness for the Macintosh developer. In addition to supporting all the major development environments on the Macintosh, QUED/M offers dozens of powerful editing features, including unlimited undo and redo, UNIX style GREP searching, macro language, scripting, text folding, text sorting, file comparison and merging, Toolbox lookup, ten editable/appendable clipboards, line numbering, markers, displaying text as ASCII codes, vertical and horizontal screen splitting, plus much more. $149

ScriptGen Pro™ by StepUp Software is an Installer script generator which requires no programming or knowledge of Rez. Supports StepUp's InstallerPack, StuffIt compression, custom packages, splash screens, network installs, Rez code output, importing resources, and AppleEvent link w/MPW: $169

SoftPolish by Language Systems is a development tool that helps software developers avoid embarrassing spelling errors, detect incorrect or incompatible resources and improve the appearance of their Macintosh software. SoftPolish examines application resources and reports potential problems to a scrolling log. Independent of any programming language or environment, SoftPolish improves the quality of any Macintosh program. $169

Spyer by InCider is a simple operated tool that records all actions (including mouse movement) you perform on a Macintosh computer and then replays them at your preferred speed. The recorded data can be saved in files for future use. Spyer works as a background process with any Macintosh application and is triggered by user defined Hot Keys Spyer enables the "Continuous Redo" utility and is especially useful for software testing and demonstration. $39

StoneTable: A library replacing all functions found in list manager plus: variable size columns/rows; different font, size, style, forecolor, backcolor per cell; sort, resize, move, copy, hide columns/rows; edit cells/titles in place; titles for columns/rows; multiple lines per cell; grid line pattern/color; greater than 32k data per table; up to 32k text per cell; support for balloon help and binary cell data. Versions for Think C, Think Pascal, MPW C, MPW Pascal, CodeWarrior C. (all prices per developer) $150 first compiler, additional compilers $50

Stone Table Extra: Additional functions for StoneTable. Drag selected cells within table or to other tables; optionally add rows as part of drag; popup menus or check boxes in cells; variable width grid lines; move/drag/resize table in window; clipboard operations on multiple cells. Requires StoneTable. (all prices per developer) $50 first compiler, additional compilers $25

StoneTable and StoneTableExtra for PowerPC: Same functionality as 68K libraries. Versions for MPW C and CodeWarrior C. Must have 68K libraries. (all prices per developer) StoneTable $100, StoneTableExtra $25

NEW! **Version Control** by Barking Dog Software Co. provides easy to use source code control for CodeWarrior, Think C, Symantec and other text files. Drag & Drop your checkins for speed. Recover any individual version or an entire release using the snapshot feature. Tracks the who, when and why of source changes and makes backing out changes a simple task. Version Control is stand-alone, requiring no other tools to operate. $199 for a single user. Multiuser licenses are; 2 users $249, 3-5 users $349, and 6-10 users $499.

NEW! **Voodoo** is a version control tool for the simple and clear management of projects in which files are created in numerous versions (variants and revisions). Voodoo allows both variant and revision control, and it manages not only variants and revisions of single files, but of a whole software project (multi files, multi users, multi variants, access rights, ...).The tool offers a neat graphical user interface and is not only suitable for mere source code control but can handle all different kinds of files with amazing compression rates: typical size of delta between arbitrary files 5% (in words: five per cent) !!!! no matter whether the files are plain text or any other documents – e. g. MSWord, 4D, Canvas, FileMaker ...).Please note special prices for multiple copies. single license – $ 190 2 pack – $ 300 5 pack – $ 665 10 pack – $ 1140 20 pack – $ 2000 Add'l pricing available on request

Want more product info? E-mail us at productinfo@xplain.com

ADVERTISER/ PRODUCT LIST

LIST OF ADVERTISERS

LIST OF PRODUCTS

TIPS & TIDBITS

By Scott T Boyd, Editor

$50 TIP OF THE MONTH

Flush Out Your Bugs

Before releasing an application to your testers, load these three extensions: EBBE (Even Better Bus Error), DisposeResource, and DoubleTrouble. They check for NIL dereferences, improperly-disposed resources, and multiply-disposed memory manager blocks. *[You can find these extensions in many places, including our ftp and other online sites – Ed stb]* Run your application. You may be surpised at what you find.

Using these debugging tools, I caught a bug in an application based on the 1/18/95 release of Sprocket. This bug was fixed in the 1/25/95 Sprocket release, and does not exist in the 12/94 release. I'm using this code both as an example and a suggested patch if you are using the 1/18 version of Sprocket.

```
TSplashWindow::~TSplashWindow()
        {
        if (fSplashPicture)
                DisposeHandle((Handle) fSplashPicture);  // Oops!
        this->Close();
        }

TWindow::Close(void)
        {
        WindowPtr       newFrontWindow = nil;
        if (FrontNonFloatingWindow() == fWindow)
          newFrontWindow = (WindowPtr)((WindowPeek)fWindow)->nextWindow;
        this->Activate(false);
        DisposeWindow(fWindow);
        if (newFrontWindow)
           HiliteAndActivateWindow(newFrontWindow,true);
        return true;
        }
```

When I ran the executable, I dropped into my low level debugger, and looked at the stack. Return addresses on the stack from the MacsBug stack crawl sc7:

```
Stack Addr  Frame Addr   Caller
 01C6ECD0               01B6D6D2 'CODE 0001 2690 Sprocket Lib'+0172
 01C6ECC8               01B6D692 'CODE 0001 2690 Sprocket Lib'+0132
 01C6ECAA    01C6ECA6   01B6E9F8 main+01F2
 01C6EC9E    01C6EC9A   01B6EF7A TSplashWindow::~TSplashWindow()+002C
 01C6EC88               01B6FA3C TWindow::Close()+0034
 01C6EC80    01C6EC7C   408578C4 _DisposeWindow+0004
 01C6EC6C               4085A846 _FrontWindow+0096
 01C6EC54               40877EB4 _KillPicture+0004
 01C6EC40               4085785C _NewWindow+010C
 01C6EC34               40877EB4 _KillPicture+0004
 01C6EC30               4085785C _NewWindow+010C
```

From this stack trace I saw that `fSplashPicture` got disposed again by `DisposeWindow(fWindow);` A review of the code showed the offending extra `Dispose` in `~TSplashWindow()`.

Why not leave these debugging resources on your system all the time? Well, it appears that not everyone uses these tools religiously and several other of my favorite (nameless) applications write to location Zero and trigger EBBE. I've sent in bug reports to the offenders, but I am still waiting for all the update releases.

Happy Bug Trails!

– Gary W. Powell
gpowell@mv.us.adobe.com

*Send us your tips or we'll install EvenBetterBusError on your machine! On the other hand, we might just pay you **$25** for each tip we use, or **$50** for Tip of the Month. You can take your award in goods, subscriptions or US$. Make sure any code compiles, and send tips (and where to mail your winnings) to all of us here at editorial@xplain.com. See page two for our other addresses.*

Powering Up The RTM Instruction

For you power programmers out there, here's a little tip on taking advantage of the PowerPC architecture. Where you might have used the 68020 RTM instruction (Return from Module), you can get a lot of mileage by using the new PowerPC RTFM instruction instead. We can assure you that you will save literally hours of time.

– David Gorkan and Allan Foster

Save a Little Time

Here is a great and easy way to supercharge File Open... File Save and File Save As... operations from any application, especially if you use the Apple Menu Items to store aliases to folders: Make an alias of the Apple Menu Items folder and put it on the Desktop.

What does that get you? Well, since the Standard file Get and Put Dialogs all have a Desktop button, you can easily get to the Apple Menu Items, and hence access any aliases that are stored there. Just Click on the Desktop folder and the Apple Menu Item Alias appears as a supercharged Volume of Aliases.

But wait, there's more! You get a real bonus from using System 7.5 because, if

Continued on page 87

SmalltalkAgents®

A Dynamic Object-Oriented Development Environment

Call now to receive our customer's own descriptions of why they're choosing AO/S to deliver their products!

Agents Object System (AO/S): Delivering Component-based Technology

SmalltalkAgents is a set of development and authoring components built on the Agents Object System™ (AO/S™). The AO/S Component Toolbox™ is a portable layer of abstraction between AO/S Components and host system services. The AO/S delivers a user extensible home/container that transparently wrappers a variety of component technologies including OpenDoc, OLE, and OSA Scripting.

AO/S: Core Technologies

The AO/S Core Technologies include rich intelligent component and agent support, vendor independent database services, concurrent application support (much like the Apple Finder), separate user interface threads enabling both tethered development and robust "memory protected" application deployment, and a shareable object system, all of which makes it ideal for high-performance client and server applications.

Scalability

Unlike other" application builders", the AO/S Product Family (including SmalltalkAgents Professional and VisualAgents) allows you to go from user-level scripting to full professional-level system development and back. SmalltalkAgents is scalable, from the creation of small and simple applications to sophisticated and complex multi-user systems.

Based on industry-proven Smalltalk language

Smalltalk is the pure Object-Oriented (OO) language which defined the OOP concept, and is now the corporate language of choice for new business applications and sophisticated client/server systems.

What this means to you!

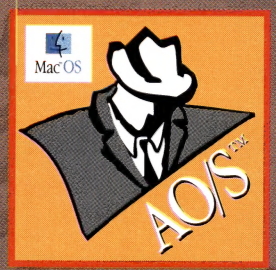
SmalltalkAgents Professional, built on the unique AO/S architecture, is a modern, innovative new development system that blends the best of component-based architecture with the best in object-oriented development technologies.

What the press and customers say:

"..... the most unique and innovative Smalltalk development environment..."
Object Magazine, October 1994

"I must say, this is NOT the Smalltalk of several years ago, and I AM TRULY IMPRESSED WITH WHAT YOU HAVE ACCOMPLISHED!"
E.S. Taylor, Independent Consultant, New York, NY

"SmalltalkAgents has been my favorite Smalltalk environment and I believe it represents the best chance Smalltalk has of becoming an accepted grass roots language for developers."
David Scott, General Atomics, San Diego, CA



Agents Object System
MACINTOSH

Don't be passed by, catch the Component Technology wave with SmalltalkAgents! To gain the Power Macintosh advantage now, call:

1-800-296-1339
or 301-530-4853 (info@qks.com)

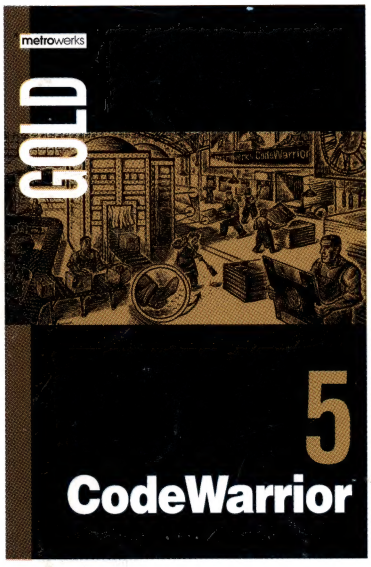
QKS

Buy now to receive free upgrade to SmalltalkAgents Pro v2.0!

Quasar Knowledge Systems • 9818 Parkwood Drive • Bethesda, Maryland 20814-4028 USA • Phone: 301-530-4853 • Fax: 301-530-5712

metrowerks® CodeWarrior™

The fastest way to write 68K and Power Macintosh code.



Twelve Compilers!
CodeWarrior Gold includes twelve compilers and CodeWarrior Bronze includes four compilers. With our Power Mac-hosted 68K cross compilers, you can build both 68K and Power Mac apps on the Power Mac. Conversely, with our 68K Mac-hosted Power PC cross compilers you can build 68K and Power Mac apps on a 68K Mac.
Note: CodeWarrior compilers are shipped as fat binaries!

Native IDEs!
The CodeWarrior integrated development environment (IDE) is native on both the 68K Mac and Power Mac. The same easy-to-use environment is used to program in C, C++ and Pascal.
Note: CodeWarrior IDEs are also fat binaries.

Get To Market Faster!
CodeWarrior compilers build your 68K and Power Mac apps at +200,000 lines/minute on a Power Mac 8100, with comparably high rates on less powerful Macs!

Two Free Updates!
CodeWarrior is updated 3 times a year on CD-ROM. Purchase either version of CodeWarrior and you are entitled to receive two future updates free of charge.

New Features!

- MPW tools included for both the 68k and Power PC
- C++ Templates supported in the compilers
- Global optimization improved for the Power PC C/C++ compilers
- Conditional breakpoints and expression evaluation added for the debugger

Two CodeWarrior versions to choose from!
Choose the CodeWarrior that suits your needs:

	BRONZE 68K Mac	GOLD Power Mac/68K Mac
Power Mac-hosted Power Mac IDE:		
Power Mac-hosted C/C++ generating PPC code		•
Power Mac-hosted Pascal generating PPC code		•
Power Mac-hosted 68K Mac IDE:		
Power Mac-hosted C/C++ generating 68K code	•	•
Power Mac-hosted Pascal generating 68K code	•	•
68K Mac-hosted Power Mac IDE:		
68K Mac-hosted C/C++ generating PPC code		•
68K Mac-hosted Pascal generating PPC code		•
68K Mac-hosted 68K Mac IDE:		
68K Mac-hosted Pascal generating 68K code	•	•
68K Mac-hosted C/C++ generating 68K code	•	•
MPW Tools:		
Power Mac-hosted C/C++ generating PPC code		•
Power Mac-hosted C/C++ generating 68K code	•	•
68K Mac-hosted C/C++ generating PPC code		•
68K Mac-hosted C/C++ generating 68K code	•	•
PowerPlant Application Framework	•	•
PowerPlant Shared Library for Power Mac		•
Source-level debugger for Power Mac		•
Source-level debugger for 68K Mac	•	•
Constructor Visual Interface Editor	•	•
Selected Apple Developer Tools	•	•
30-day money-back guarantee	•	•
3,000 pages of On-line Documentation and much, much more.	•	•

1994 BYTE AWARD OF EXCELLENCE

CodeWarrior 5 Gold
For Power Macintosh & 68K Macintosh development.
LNG 0070.....$399

CodeWarrior 5 Bronze
For 68K Macintosh development.
LNG 0068.....$99

Requires: Motorola 68020 or higher or PowerPC 601 processor; 8MB RAM; System 7.1; CD-ROM Drive

metrowerks®

CALL 1-800-377-5416
International (419) 281-1802
Fax (419) 281-6883